12-2017

Hierarchical Off-Road Path Planning and Its Validation Using a Scaled Autonomous Car'

Angshuman Goswami
Clemson University

Hierarchical Off-Road Path Planning and its Validation using A Scaled Autonomous Car

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mechanical Engineering

by
Angshuman Goswami
December 2017

Abstract

In the last few years. while a lot of research effort has been spent on autonomous vehicle navigation, primarily focused on on-road vehicles, off-road path planning still presents new challenges. Path planning for an autonomous ground vehicle over a large horizon in an unstructured environment when high-resolution a-priori information is available, is still very much an open problem due to the computations involved. Localization and control of an autonomous vehicle and how the control algorithms interact with the path planner is a complex task. The first part of this research details the development of a path decision support tool for off-road application implementing a novel hierarchical path planning framework and verification in a simulation environment. To mimic real world issues, like communication delay, sensor noise, modeling error, etc., it was important that we validate the framework in a real environment. In the second part of the research, development of a scaled autonomous car as part of a real experimental environment is discussed which provides a compromise between cost as well as implementation complexities compared to a full-scale car. The third part of the research, explains the development of a vehicle-in-loop (VIL) environment with demo examples to illustrate the utility of such a platform.

Our proposed path planning algorithm mitigates the challenge of high computational cost to find the optimal path over a large scale high-resolution map. A global path planner runs in a centralized server and uses Dynamic Programming (DP) with coarse information to create an optimal cost grid. A local path planner utilizes Model Predictive Control (MPC), running on-board, using the cost map along with high-resolution information (available via various sensors as well as V2V communication) to generate the local optimal path. Such an approach ensures the MPC follows a global optimal path while being locally optimal. A central server efficiently creates and updates route critical information available via vehicle-to-infrastructure(V2X) communication while using the same to update the prescribed global cost grid.

For localization of the scaled car, a three-axis inertial measurement unit (IMU), wheel encoders, a global positioning system (GPS) unit and a mono-camera are mounted. Drift in IMU is one of the major issues which we addressed in this research besides developing a low-level controller which helped in implementing the MPC in a constrained computational environment. Using a camera and tire edge detection algorithm we have developed an online steering angle measurement package as well as a steering angle estimation algorithm to be utilized in case of low computational resources.

We wanted to study the impact of connectivity on a fleet of vehicles running in off-road terrain. It is costly as well as time consuming to run all real vehicles. Also some scenarios are difficult to recreate in real but need a simulation environment. So we have developed a vehicle-in-loop (VIL) platform using a VIL simulator, a central server and the real scaled car to combine the advantages of both real and simulation environment. As a demo example to illustrate the utility of VIL platform, we have simulated an animal crossing scenario and analyze how our obstacle avoidance algorithms performs under different conditions. In the future it will help us to analyze the impact of connectivity on platoons moving in off-road terrain. For the vehicle-in-loop environment, we have used JavaScript Object Notation (JSON) data format for information exchange using User Datagram Protocol (UDP) for implementing Vehicle-to-Vehicle (V2V) and MySQL server for Vehicle-to-Infrastructure (V2I) communication.

Dedication

To my parents, teachers, guides, Puja, my brother, Mamu and all the people who have inspired me in different walks of life.

Acknowledgments

I would like to thank my family and friends for their unconditional support and strong backing during the course of this research without which it would not have been possible. I would like to thank Prof. Vahidi for believing in me and providing me the opportunity to be a part of this research project. His guidance, attention to detail and enthusiasm towards the project has helped me during all the stages of the research. I would also like to thank my committee members, Prof. Tallapragada and Prof. Wagner for their support to this research. I would like to thank Dr. Nianfeng Wan for his support in developing the programming tool, Dr. Judhajit Roy for his support in validating our work in a simulation environment, Austin Dollar for his suggestions in many crucial points of this research, Dr. Ali Fayazi for his help on vehicle communications, Dr. Hamed Saeidi for his help in image processing and Matthew Chicky for his help in building the scaled car.

I would like to thank Dr. Paramsothy Jayakumar and the US Army TARDEC and Automotive Research Center at University of Michigan for their technical and financial support of this research. I would also like to thank Dr. Chen Zhang from Ford Motor Co. for providing an industrial perspective on this research project.

Table of Contents

List of Figures

Chapter 1

Introduction

The field of off-road navigation has not yet been explored as exhaustively as on-road navigation. While this presents an opportunity for research, there are associated challenges like lack of awareness, which can not only lead to vulnerable situations but also increased energy usage. Uninformed decisions in such terrain can cause loss of a vehicle. Advances in connected vehicle technologies and real-time access to computational clouds have caused a paradigm shift in the efficiency of fleet management. Connected vehicles which can talk amongst themselves as well as servers, exchange information, and automatically learn from each other's trip experience. Availability of precise 3D elevation profiles, soil trafficability, and vegetation maps present an opportunity for further improvement in off-road navigation. Off-road driving presents a significantly different set of challenges when compared to on-road driving [10,18,35,39]. In off-road vehicles, more route considerations for the optimization routines needs to be taken since they are not constrained to the road network. Efforts have been made for improving ride quality in off-road driving based on terrain roughness [38]. In recent years, extensive efforts in the field of detection, classification and mapping of terrain can largely be classified into two approaches; i) developing a terrain model with obstacle constraints based on vehicle sensors and ii) using high-resolution terrain information collected a priori [7,10,24,28]. For generating the optimal route over a given terrain, an opportunity exists to utilize both sources of data to better inform the decision process.

A part of our research involves the investigation of an efficient hierarchical approach for vehicle path planning across vast off-road terrains. A global optimization routine utilizes low resolution a-priori information like terrain, soil and visibility to find an optimal cost-to-go grid. This approach helps in distributing computation cost between the on-board computer and the cloud. A global path planning algorithm can run in the cloud, updating the cost-to-go grid based on the augmented information available from V2X communications. A local optimization routine incorporates high-resolution information and builds on the cost map, arriving at a global optimal path. The cost-to-go grid ensures that the local planner is aware of the optimal trajectory in case it deviates from the mandated path. In our simulation, we have utilized publicly available terrain and soil maps in our algorithms before running vehicle-in-the-loop testing. Improving safety of vehicles is one of the prime motivations for increasing autonomy. In military applications, visibility of a platoon from the nearest communication towers or avoiding line-of-sight of adversarial observation towers is of paramount importance. We address those objectives in our path planning algorithm. We will use the terrain maps and known positions of adversarial towers, to generate a visibility map and incorporate it into our optimization routine. For remote fleet operation, assisting the vehicle operator with updated route guidance can make a critical difference.

After algorithm verification in a simulation environment, we built a scaled car to validate our algorithm in a vehicle-in-the-loop environment. In our proposed scenarios, a real vehicle communicates with virtual vehicles and a centralized server running on a back-end computational cloud. The system relies on recent advances in vehicular connectivity that enable individual vehicles to cooperate and exchange information within a fleet and to communicate with back-end infrastructure. The setup provides an affordable research testbed which aims to reduce the costs associated with the use of full-scale autonomous vehicles. Such monetary and time costs restrict many researchers to pure simulation environments [32] where it is difficult to mimic real world issues like communication delay, sensor measurement noise and errors, and modelling error in vehicle dynamics. Many researchers have worked on the development of a scaled-down platform which provides a balance between full-scale vehicles and pure computer simulations. La et. al. [22] have developed a scaled-down platform for intelligent transportation systems (ITS) that is useful for preliminary study and feasibility tests. Verma et. al. [41] have designed a scaled vehicle with longitudinal dynamics of a high-mobility multi-purpose wheeled vehicle (HMMWV). Travis et. al [40] have worked on using scaled vehicles to investigate rollover propensity. For autonomous operation and navigation of a scaled car, its ability to localize itself is critical. Considerable efforts have been made in the area of simultaneous localization and mapping (SLAM) [23] with high success in indoor environments [15]. Stereo vision has been used with inexpensive GPS [5] for localization in outdoor environments while Kalman Filter based integration with GPS, IMU, odometry and Lidar measurements has also been explored extensively [31].

While many advancements have been made in this domain, there exists a scope for development of a low-cost platform implementing feedback steering control and Vehicle-to-Everything (V2X) communication using low-cost sensors. We have built a scaled car for our experiments based on Berkeley Autonomous Race Car (BARC) [2], a low-cost open-source development platform for autonomous driving developed at University of California, Berkeley. We have improved on the base software of the BARC platform, so that we can use it for our preliminary study and feasibility tests. We have worked on running this car on a flat terrain (which reduces the complexity of all computations to a 2-D domain) while our future work will consider implementation in 3-D terrains. To simulate a fleet and analyze the impact of connectivity, we have introduced V2X communication via a wireless network between the real car and virtual vehicles running on a back-end cloud. This enables individual vehicles to cooperate and exchange information within a fleet and to communicate with a back-end server.

Chapter 2

Hierarchical Route Guidance Framework

2.1 Architecture

The proposed hierarchical route guidance approach seeks to address the challenge of high computational cost when finding the optimal path over large terrain when high-resolution information is available. On-road applications typically optimize the route based on existing known road networks. In off-road applications, no such pre-defined road network exists which makes the optimization task even more challenging.

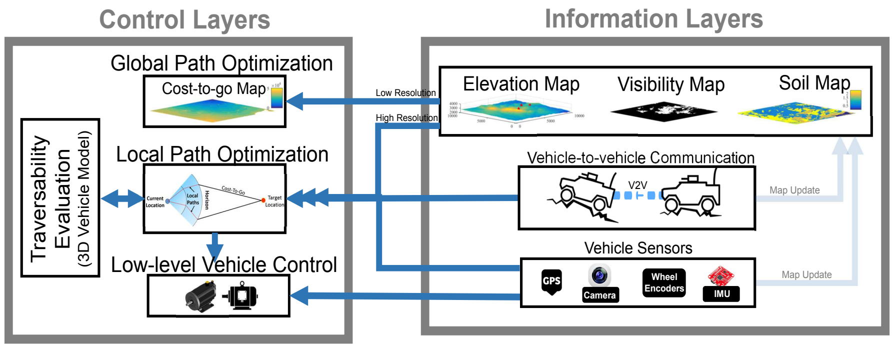


Figure 2.1: Overview of the route guidance framework.

An overview of the software architecture and a flowchart of the developed hierarchical approach is shown in Figure 2.1. A dynamic programming routine is used to generate the approximate global optimal path and the optimal cost-to-go map using low resolution elevation and soil trafficability information. The optimal cost-to-go map is a map whose value at each grid point is the optimal cost-to-go from that grid point to the target destination. It is discussed in more detail in Section 3.1. A model predictive local optimization routine uses the stored optimal cost-to-go map and high-resolution information from the on board sensor to find the local optimal path. A part of the ongoing research in our group is traversability evaluation using a high order 3D vehicle model which is not a part of this thesis.

A typical multiple level optimization will calculate an optimized trajectory followed by a low level control. The optimized trajectory from high level may not be updated in real-time due to computational cost especially for long term/large scope planning. Dynamically calculating the optimal path for a fleet of vehicles with the same destination, but different real-time position may be more challenging. The cost-to-go based method does not need to calculate the optimal path globally or need memory to save the path. The backward based cost-to-go could be used by all vehicles at different locations as long as they have the same target position. An offline optimization routine uses coarse resolution information to generate an approximate global optimal path. Each ground vehicle performs online optimization to find the optimal local path using high-resolution information. The optimization routine uses a back-end server to obtain, store and share relevant information amongst them.

2.2 Information Layers

An efficient path planning algorithm is predicated on efficient handling of the information database. An novel method of integrating the information layers to the decision process is presented here. Information collected before and during the mission is stored as maps. Multiple maps, each representing an information layer, are stored on a backend server. Information stored on the server is accessible by the vehicles on the ground as well as by a central computation server. The route guidance routines use these stored maps to perform their respective optimizations. All the maps are stored with a corresponding timestamp representing the latest update. The maps are categorized into two types: i) Static maps represent information layers that remain constant over the full duration of the mission, ii) Dynamic maps contain information layers that are either updated or generated during the mission. Examples of the maps we have used in this work are shown in Figure 2.2.

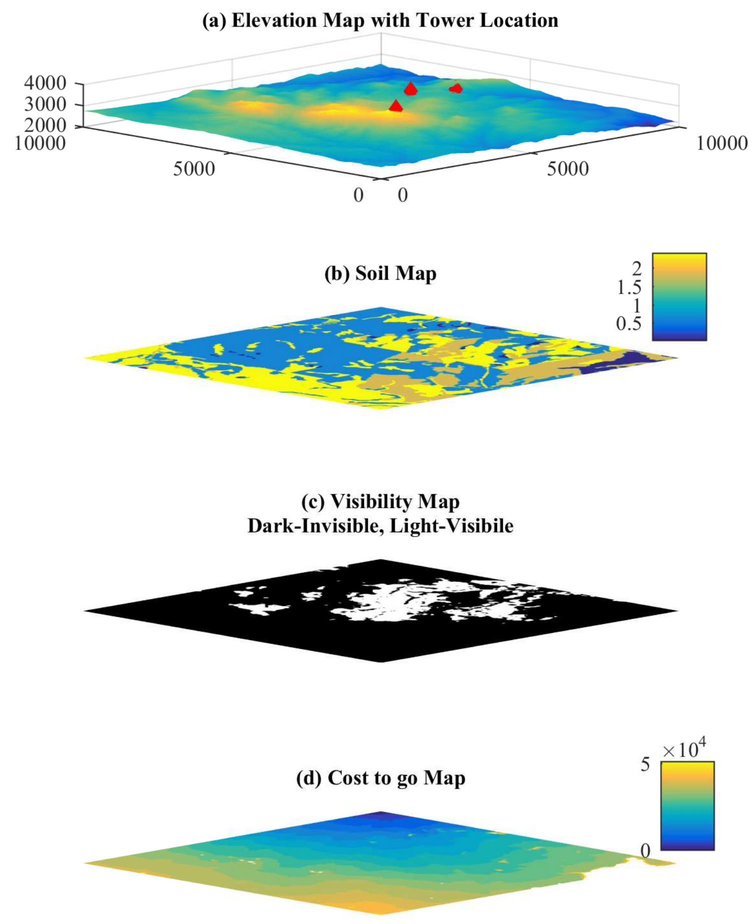


Figure 2.2: Information Layers (Figure adopted from [34])

- **Elevation Information:** Various Digital Terrain Models (DTM) exist that provide topographic elevation data of bare terrain. Over the past 15 years, the elevation information of the United States National Map [3] was mapped by United States Geological Survey (USGS) and presented in the National Elevation Dataset (NED). The NED bare earth elevation information for the entire US is available for public download at medium resolution (every 10 or 30 meters). Since 2015, under its more recent 3D Elevation Program (3DEP), USGS now provides higher resolution elevation data (every 1/9 arc-second and 1 meter) for selected locations in the US and plans to expand this dataset in the coming year [1]. We have selected a representative site where the higher resolution data is available and have successfully retrieved the elevation. For our study, we chose a 10 sq. km. area near Denver, Colorado. This area has a rough terrain with two peaks, captured with high 1m elevation resolution as shown in Figure 2.2a.

Other existing DTM elevation sources are through the Shuttle Radar Topographic Mapping (SRTM, every 30 m) and Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER, every 30 meter) programs available by NASA and Japan's Space Agency. Intermap Technologies provides high-resolution data (every 5 meter) for a wide range of locations across the world.

- **Soil Trafficability:** The US Department of Agriculture (USDA) publishes soil information for the entire contiguous USA [4]. Among the publicly available data, soil trafficability is the most pertinent information when planning for off-road driving. The soil trafficability layer is intended to rate the capacity of the soil to support military vehicles. Soil trafficability ratings account for soil strength, soil slipperiness, stickiness, stoniness, and slope. The rating class indicates the verbal description of the expectation to complete the task for the given military vehicle category, number of passes to complete the task, and the season of the year. Military vehicles are categorized into seven types for trafficability rating. Table 2.1 shows various trafficability ratings and the expected task completion value. A manual [6] provides a guideline to convert the verbal soil rating to a probabilistic task completion value as indicated in the second column of Table 2.1. The route guidance routine has been developed for deterministic maps. Therefore, the probabilistic task completion values are converted to corresponding numeric values for computational ease as shown in the third column of Table 2.1.

Table 2.1: Soil Trafficability Ratings.

Rating	Expected Completion Rate	Our Rating
Excellent	0.90–1.00	4
Good	0.75–0.89	3
Fair	0.50-0.74	2
Poor	0.00-0.49	1
Not Rated	-	0.01

To run the optimization routines for the area corresponding to terrain map in Figure 2.2a, we have extracted the soil trafficability data from the USDA database as shown in Fig. 2.2b.

- **Line of Sight (LoS):** The concept of line-of-sight was initially used in guided missile development in the 1940's [8]. Recent efforts in tactical path-finding use the concept wherein safety of the unit is taken into account in path optimization by being out of the LoS or enemy range while remaining in the friendly LoS [20, 21, 25, 36]. In mountainous or hilly terrain, there are issues related to failure of radio communication. In such situations, the concept of LoS can be extended to find the optimal path keeping a fleet within range of the radio tower of the base stations effectively improving safety of the operation. Steve et al. [9] have explored a voxel-based modelling approach for rapid viewshed calculation. A viewshed is the geographical area that is visible from a location which includes all surrounding points that are in line-of-sight with that location and excludes points that are beyond the horizon or obstructed by terrain and other features like buildings, trees. (source : Wikipedia). Here we have used the line of sight as an extra information layer which aids in the trajectory optimization. The elevation layer is used to find the grids which are visible from known locations of adversarial watch towers.

Figure 2.2c shows areas in the map which are visible from the known location of three fixed towers in white. Darkly shaded areas are invisible parts of the terrain. The location of the towers are shown in Figure 2.2a with red markers. MATLAB function *voxelviewshed* was used to translate the elevation information and the tower location to the visibility map.

Chapter 3

Optimal Off-Road Navigation

We initially developed a simulation environment implementing the framework discussed in Chapter 2 in Matlab using a global and a local planner. We had run initial simulations to validate the effectiveness of the developed hierarchical algorithm. The simulation was run for the 10×10 kilometre terrain shown in Figure 2.2. The objective of the mission was to travel optimally from the start point at (0,0) to the end point at (10000, 10000) meters. The research carried out in this Chapter has contribution of Dr. Judhajit Roy was had formulated the MPC and the simulation environment, Nianfeng Wan who had developed the global planner. My specific contribution is the developing of line of sight (visibility), MPC formulated including the line of sight and the system in the hierarchical framework utilizing a centralized server developed using MySQL. This chapter has been discussed in our research [34].

3.1 Global Planner

Dynamic Programming (DP) is a common and powerful method in solving optimization problems. Based on Bellman's principle of optimality, DP divides the whole optimization problem into smaller sub-problems and solves them recursively. It also stores the intermediate results, thus helping in solving sub-problems in case they reoccurs, thereby significantly reducing computational time with modest expenditure in storage space. In computer science literature, Dijkstra's algorithm [11] is one of the most important and useful methods in solving a path planning/shortest path problem. The algorithm normally utilizes an environment represented as a graph with vertices and edges to finds the minimum cost (e.g length) between each vertex and the given vertex on the graph. Fibonacci heap implementation in the algorithm makes it the

Dr. Judhajit Roy contributed in formulation of the MPC and simulations and Dr. Nianfeng Wan contributed in development of the global planner. [34].

fastest single source path finding algorithm for non-negative cost, asymptotically [13].

Dijkstra's Algorithm was usually described as a greedy algorithm before 2000s, in the computer science and operational research fields. The connection between Dynamic Programming and Dijkstra's algorithm was successfully shown by Sniedovich [37]. Dynamic Programming is usually used in research where cost-to-go approximation is required to be obtained from all grids to the destination. Also with advanced data structure implementation, Dijkstra's Algorithm can achieve better computation performance with less storage expense. Hence, we have chosen Dijkstra's algorithm for our global planning.

In our research, the entire terrain as shown in Figure 2.2 (a), is divided into small square grids and each grid representing a vertex on a graph. Each small grid is connected to only its 8 neighbouring grids. The cost of a vehicle travelling from a grid i to j is defined by a cost function. The edges are assumed to be undirected, which means travelling in the reverse, i.e. from grid j to i, cost remains the same. While Dijkstra's algorithm is mostly used to find the shortest route or distance, here we have increased the scope of the algorithm by including different information layers such as distance, elevation changes and soil conditions into account. In our research, the cost function is defined as follows:

$$\begin{aligned} J(i,j) &= \sum_{k=1}^{n} w_k J_k(i,j) \\ J_1(i,j) &= \sqrt{(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2} \\ J_2(i,j) &= \frac{1}{s_i}+\frac{1}{s_j} \\ J_3(i,j) &= |z_i-z_j| \end{aligned} \tag{3.1}$$

where x,y,z are the 3D coordinates of the grid, s is the trafficability coefficient and w is the weight coefficient. J_1, J_2 and J_3 represents the cost due to distance travelled, soil condition and absolute elevation change respectively. DP finds the path which minimizes the total cost. Including J_2 in the cost function ensures that the route with better soil condition is chosen. J_3 ensures that the route chosen has smaller crest and trough. To avoid extreme slope which are beyond the driving capability of the vehicle, the following constraint is imposed in the optimization.

$$slope(i,j) < slope_M AX \tag{3.2}$$

While travelling from one grid to another, if the slope constraint is violated, the cost is set to infinity. The cost function can be expanded to incorporate more information layers with different associated weights which can be tuned based on objective of the mission.

The cost-to-go map is generated as the output of Dijkstra's algorithm which computes the minimum cost incurred when travelling from any grid on the graph to the final destination grid.The choice of resolution or the size of each grid determines the computation complexity and the storage space needed. The grid resolution is directly proportional to the memory usage while indirectly proportional to the precision. Hence, with an optimal resolution of 50 m is chosen, balancing precision and required memory. Each grid is considered as one vertex and, different elevation changes or soil conditions within the grid are neglected.

The global path planning is solved off-line and the cost-to-go map is updated on the server. This cost-to-go map is utilized by the local path planning along with high resolution and other available information to find the global optimal path.

3.2 Local Route Guidance via Model Predictive Path Planning

The lower level route planning is assisted by the cost-to-go map generated in the high-level route planning which helps us achieve efficient integration of global and local planner. Model predictive control (MPC) with receding horizon approach is used for the local route guidance as the vehicle navigates though the terrain. For the MPC, we have used a simplified kinematic model of the vehicle defined by 3 states: x and y are the coordinates of the center of gravity with respect to vehicle's initial position and ψ is the vehicle heading angle with respect to the initial longitudinal direction. The input to this model is the commanded change in heading angle and is denoted by δ. The discretized model is given by,

$$\begin{aligned} x_{k+1} &= x_k + \Delta s_k \cos(\psi_k) \\ y_{k+1} &= y_k + \Delta s_k \sin(\psi_k) \\ \psi_{k+1} &= \psi_k + \delta_k \end{aligned} \tag{3.3}$$

where Δs_k is the the displacement of the center of gravity in the x-y plane over step k. For our simulations, we have used the same model for the plant. In Section 4.9.2 we will see how the commanded heading angle δ can be utilised by a low level vehicle controller for steering control to meet the desired heading angle.

Typically, MPC used for path planning and tracking [33, 42] employs a fixed temporal horizon. In our research, we have used a fixed spatial horizon that encircles the vehicle as shown in Figure 3.1 for our model predictive algorithm to run the local route guidance. A circular horizon is consistent with the vehicle sensors horizon and therefore simplifies the formulation. Accordingly, discretization in Equation 3.3 and in

Dr. Judhajit Roy contributed in formulation of the MPC and simulations and Dr. Nianfeng Wan contributed in development of the global planner. [34].

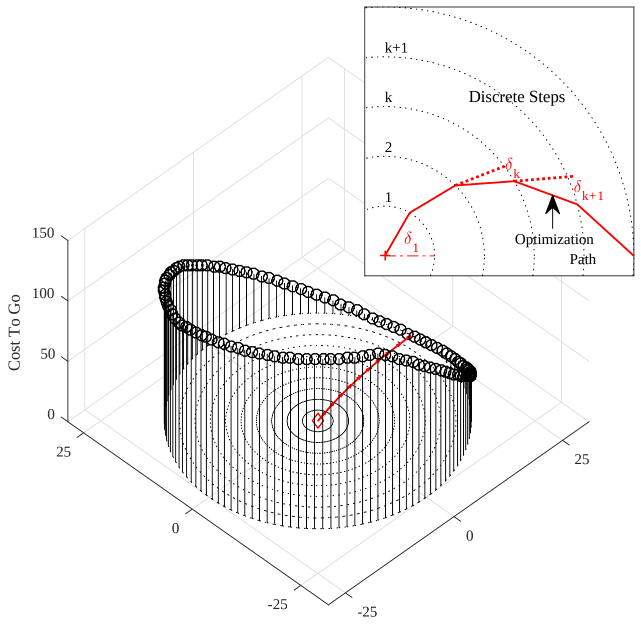


Figure 3.1: MPC radial steps and the optimal cost-to-go at the end of MPC circular horizon shown as varying fence height (Figure adopted from [34]).

the rest of this section is over position and not over time. More specifically, the circular horizon is divided into concentric circles centered at the current vehicle location. The increase in radii, Δr, of successive circles is fixed as the MPC's step length. The steps are incremented in the x-y plane and because the vehicle is constrained to move on prescribed terrain, the change in elevation in the z direction is a function of the vehicle's x-y location.

As schematically illustrated in Figure 3.1, the MPC optimizes the local path (shown in red) based on i) high-resolution map and sensor information over its circular horizon and ii) the DP cost-to-go from the perimeter of the horizon to destination point (represented by the varying fence heights). The optimization variables are commanded change in heading angle δ_k at each Δr increment. Hard symmetric constraints are imposed on δ_k and its rate of change over each step to more accurately represent the physical constraints on the vehicle motion:

$$\begin{aligned} |\delta_k| &\leqslant \delta_{max} \\ |\delta_{k+1} - \delta_k| &\leqslant \Delta\delta_{max} \end{aligned} \tag{3.4}$$

The structure of the cost function of the local routine echoes that of the global routine for penalty surface continuity. The objective of the optimization routine is to:

$$\min_{\delta_1 \cdots \delta_n} J = w_1 \sum_{k=1}^{n} \Delta l_k + w_2 \sum_{k=1}^{n} \frac{1}{soil_k} + w_3 \sum_{k=1}^{n} |z_k - z_{k-1}| + w_4 \sum_{k=1}^{n} obstacle_k + w_5 \sum_{k=1}^{n} visibility_k + J^*_{DP}(x_n, y_n) \tag{3.5}$$

In the above cost function, n is the prediction horizon in radial steps and δ_i are the free optimization variables at successive radial steps; each represent the commanded change in heading angle at stage i. The penalty weights are design variables shown as w_i. The six terms of the cost functions are,

- Δl_k is the 3D distance covered in the k^{th} step which is computed as,

$$\Delta l_k = \|\Delta s_k, (z_k - z_{k-1})\|_2 \tag{3.6}$$

  where Δs is the topographic distance (2D), and z is the elevation at each step point.

- The function $soil_k$ is the computed value associated with the terrain traversed as described in section 2.2. Since the objective is to minimize the cost function, the inverse of the soil computed value is used in the cost function to ensure that the prescribed route utilizes the best soil possible.

- The term $|z_k - z_{k-1}|$ penalizes the elevation change in one radial step to prevent selection of steep paths. A hard constraint on $\frac{|z_k - z_{k-1}|}{\Delta s_k}$ can also be imposed to put an upper limit to the path steepness.

- To ensure that the local route prescribed avoids collision with the obstacles we penalize the locality of the detected obstacles heavily. For an obstacle q, let A(q) denote the circular area surrounding the obstacle centered at the geometric center of the obstacle q and whose circular radius is equal to the preferred safe distance between the vehicle and obstacle. The function $obstacle_k$ in Equation (3.5) is defined as,

$$obstacle_k = \begin{cases} 0 & \text{if} \quad x_k, y_k \notin \bigcup_{q=1}^{Q} A(q), \\ M & \text{if} \quad x_k, y_k \in \bigcup_{q=1}^{Q} A(q), \end{cases} \tag{3.7}$$

where M represents a very large number.

- To reduce the chance of being detected, the cost function (3.5) penalizes areas which can be seen from the adversarial towers by introducing the cost term,

$$visibility_k = \begin{cases} 0 & \text{if} \quad x_k, y_k \in invisible, \\ 1 & \text{if} \quad x_k, y_k \in visible. \end{cases} \tag{3.8}$$

- The last term $J^*_{DP}(x_n, y_n)$ represents the optimal cost-to-go, calculated by the dynamic program, from the end of the MPC horizon to the final target. This cost is schematically shown as fence heights in Figure 3.1. Remember that the values for the optimal cost-to-go were obtained (offline) using a coarse dynamic program and stored as a layer of information maps as shown in Figure 2.2d. Inclusion of the cost-to-go is intended to prevent short-sighted decisions by the local path planner and ensures local decisions are guided by global objectives.

The weights on elevation change (w_3), obstacle detection (w_4), and the line of sight (w_5) are used as soft constraints, i.e. if the respective parameter values are greater than a predetermined limit a high penalty is applied to that section of the path. The cost function in (3.5) is minimized subject to the vehicle kinematic equations in (3.3) and the input constraints in (3.4). This problem is solved numerically using a sequential quadratic programming approach. Ideally, the cost function for MPC and DP should be the same so that the costs match at the boundary of the local prediction horizon. Note that as Equation (3.5) shows, the ultimate cost that each vehicle minimizes is broken down to a shorter horizon MPC cost with higher resolution sensory information added with a longer DP horizon and lower resolution map information). The only difference between them is the resolution of the available information. Note that because the cost function in (3.5) does not explicitly depend on control inputs δ_k, in the numerical routine we minimize it with respect to heading

Dr. Judhajit Roy contributed in formulation of the MPC and simulations and Dr. Nianfeng Wan contributed in development of the global planner. [34].

angles $\psi_1, \cdots, \psi_k$ while still enforcing input constraints in (3.4); this simplifies execution of the non-linear optimization routine.

3.3 Simulation Results

As previously discussed, we ran simulations for the 10×10 kilometre terrain shown in Figure 2.2. The objective of the mission was to travel optimally from the start point at (0,0) to the end point at (10000, 10000) meters.

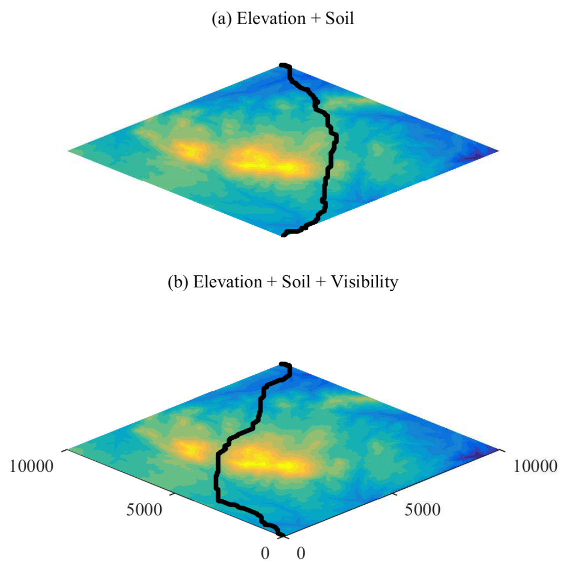


Figure 3.2: Prescribed routes with different information layers (Figure adopted from [34]).

First simulations were conducted with only one vehicle. The DP algorithm is executed off-line and then the local MPC path planner is run and is informed by the DP-calculated optimal cost-to-go map. Multiple layers of information were considered in path planning decisions by both DP and MPC. The local routine at each step tries to navigate the vehicle to the lowest cost within its horizon as well as incorporates the high-resolution map information. The global optimality of the prescribed route is enriched since the local route guidance routine augments the results of the global route optimization routine through the cost-to-go map.

The route prescribed by the route guidance routine is plotted over the terrain contour map in Figure 3.2. In sub-plot (*a*) of Figure 3.2, both terrain and soil conditions are considered in addition to soft and hard constraints on stepwise elevation change. Furthermore, command heading angle constraints shown in Equation (3.4) are also enforced. The chosen path balances travelled distance, road steepness, soil conditions, and high-level vehicle kinematic limitations. As a result, it chooses a relatively flat valley that goes around steep slopes.

An optimal path prescribed in the presence of visibility information is shown in Figure 3.2b. This reflects a scenario where it is desirable to stay out of the line of sight, e.g. of adversarial watch towers. In Figure 2.2a, we have shown 3 watch tower locations with red markers and used the MATLAB toolbox *Viewshed* to find areas which are visible from those watch towers taking into account the terrain information we already have. The visible portions in the map from the known watch tower locations are marked in white in Figure 2.2c while the invisible portions are marked in black. To remain in invisible areas, the optimal path in Figure 3.2b takes a longer stealth route passing in between the two peaks.

A fleet of three identical vehicles is considered to demonstrate the effectiveness of the connectivity for a fleet of vehicles. The neighbouring vehicles were initially separated by a equilibrium distance $R_{horizon}$, heading due east (to the right). J_{CON} described in Equation 3.9, is the cost associated to implement the connectivity between vehicles which is added to the cost described in Equation 3.5.

$$\begin{aligned} J_{CON}(p) &= w_6 \| J_{CON}(p|p-1), J_{CON}(p|p+1) \| \\ &= w_6 \| \left(L_{p|p-1} - 2R_{horizon}\right), \left(L_{p|p+1} - 2R_{horizon}\right) \|, \end{aligned} \tag{3.9}$$

where $L_{p|p-1}$ and $R_{horizon}$ represents the distance between two neighbouring vehicles and equilibrium distance respectively. Two scenarios were simulated: In the first scenario, the motion of the three vehicles are not coordinated as shown in the Figure 3.3a, and in the second case the motions of the vehicles are coordinated as shown in the Figure 3.3b. To highlight the difference, the initial 500 m$\times$500 m subsection is shown here. For both scenarios, all the vehicles eventually reach the same destination point.

We can observe that in the first scenario each vehicle's local guidance routine determines the optimal path for the vehicle without considering the location of the other vehicles in the fleet. The three vehicles have different starting points. Therefore, their optimal paths to the destination are different. The optimal path of one of the vehicles (solid route) would lead it to lose contact with the other vehicles in the fleet. Such a scenario might increase the risk to the vehicle. On the other hand, if the local path planner incorporates the location of its neighbouring vehicles, the motion of the entire fleet is coordinated. The benefit of a

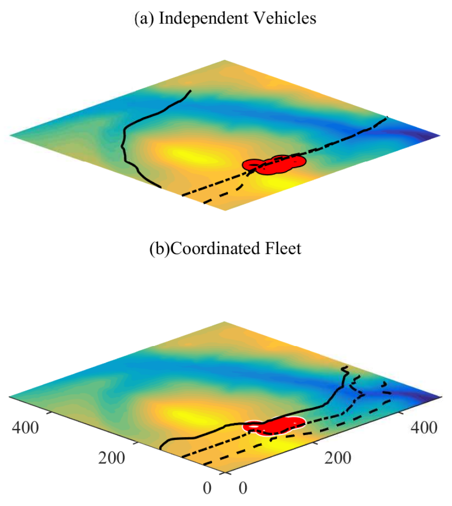


Figure 3.3: Coordinated versus individual path planning (Figure adopted from [34]).

coordinated fleet from an information resource point of view, is the locally contiguous enrichment of the information layers. On observing obstacles or steep elevation profiles with higher cost than the coordination cost, the vehicle's local path planner prescribes a path avoiding those grid points. When the risk reduces in the local vicinity, the local routines coordinate to bring the fleet vehicles back into a formation. Such a coordinated fleet behaviour would mitigate the risk for the entire fleet since the fleet horizon is extended while travelling in a formation. The flexibility of the proposed route guidance framework was demonstrated with these simulations. The fleet operators can use the developed hierarchical framework for more complex scenarios by modifying the cost functions used at each level as per their requirements.

Chapter 4

Scaled Experimental Environment

4.1 Experimental Setup

Once our optimal off-road navigation was validated in a simulation environment, we wanted to do the same in an experimental environment which will help us in testing various real world scenarios in a vehicle-in-loop platform (discussed in Chapter 6). We have build a scaled car which is controlled using a micro-controller and a host of sensors as shown in Fig. 4.1 based on the BARC platform. The steering (M_1) and acceleration (M_2) servo motors are powered by a 7.4 V battery and controlled via Arduino Nano. An Odroid XU4 forms the brain of the car powered by 2 GB RAM and 64 GB flash memory. ROS (Robot Operating System) is used as a structured communications layer above the host operating system (Ubuntu 14.04, Trusty Tahr). The low-cost suite of sensors include a triple-axis IMU (Inertial Measurement Unit) used for measuring the instantaneous yaw angle and rate, hall-effect sensor mounted on the wheel hub for vehicle speed measurement, GPS for global positioning, and a two mega-pixel mono-camera for steering wheel angle measurement. While the Encoders, IMU and GPS are used for localization (refer Sec. 4.8), the camera is used for steering angle measurement (refer Sec. 4.6.3). Four magnets are mounted on each wheel for measuring the wheel speeds using a hall effect sensor. (explained in detail in Sec. 4.7).

4.2 Vehicle Model

We use a 3-DOF and 6 state bicycle model derived from standard rigid body mechanics illustrated in Fig. 4.2. This model simplifies the vehicle dynamics as a rigid body with lumped rear and front wheels

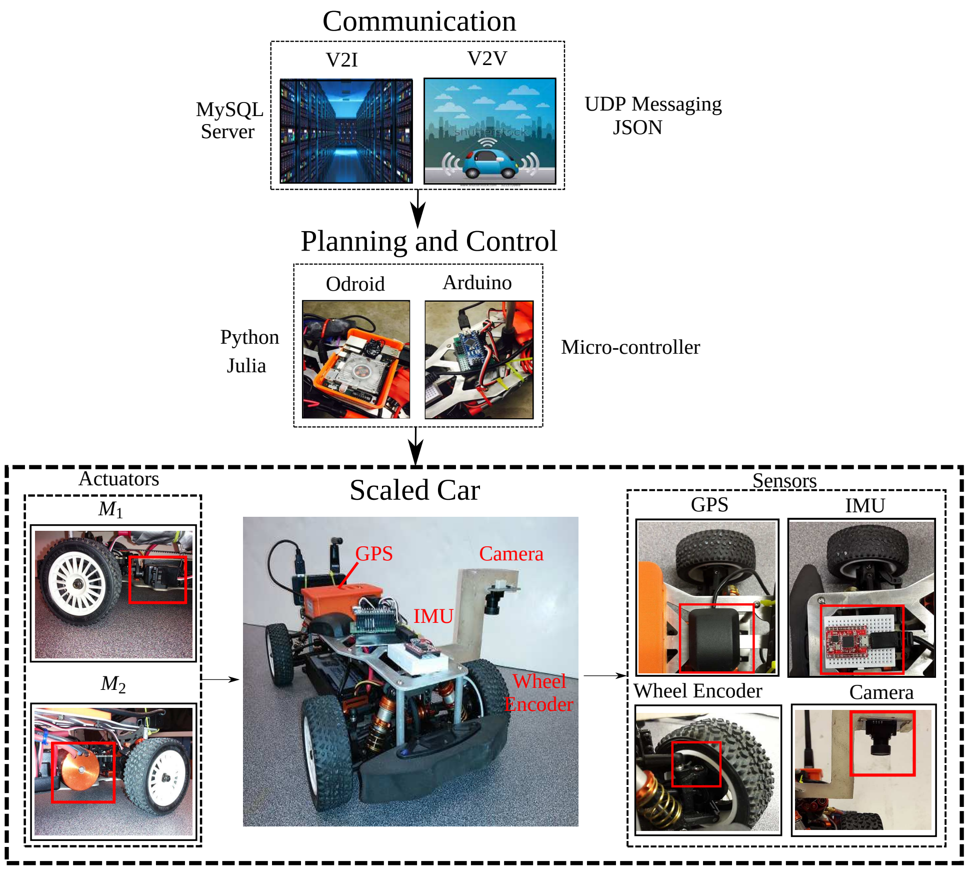


Figure 4.1: Overview of the experimental test bed

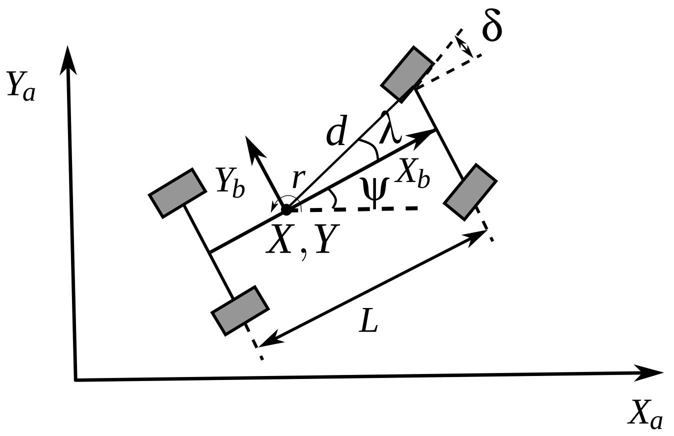


Figure 4.2: 3 Degree of Freedom (DOF), 6 state Bicycle Model.

[14, 17, 19, 29]. We neglect roll, pitch and lateral air resistance. In this figure, (X,Y) represent the coordinates of the Center of Gravity (COG) of the vehicle, ψ denotes the yaw angle and r is the yaw rate in the inertial frame (X_a, Y_a). δ is the steering angle in the non-inertial body frame (X_b, Y_b). L is the vehicle's length. The dynamic equations of the motion take the following form [19],

$$\begin{aligned}
\dot{X} &= v_x \cos\psi - v_y \sin\psi \\
\dot{Y} &= v_x \sin\psi + v_y \cos\psi \\
\dot{\psi} &= r \\
\dot{v}_x &= \frac{1}{m}\left(F_{xR} - F_{yF}\sin\delta\right) + v_y r \\
\dot{v}_y &= \frac{1}{m}\left(F_{yF}\cos\delta + F_{yR}\right) - v_x r \\
\dot{r} &= \frac{1}{I_z}\left(L_f F_{yF}\cos\delta - L_r F_{yR}\right),
\end{aligned} \tag{4.1}$$

where v_x and v_y denote longitudinal and lateral velocities, F_{xR} and F_{yR} are longitudinal and lateral forces on the rear wheels and F_{xF} and F_{yF} are their counterparts for the front wheel in the non-inertial body frame (X_b, Y_b). I_z is the moment of inertia around z axis and m represents vehicle's mass. L_f and L_r represent the distance of front and rear wheel axles from the CoG of the car. The Pacejka model [30] is used for the estimation of tire lateral forces via a function of tire slip angles.

$$F_y(\alpha) = \begin{cases} \mu F_z \sin(C\tan^{-1}(B\alpha)) & : |\alpha| \le \alpha_{cr} \\ -F_y^{\max}\mathrm{sgn}(\alpha) & : |\alpha| > \alpha_{cr} \end{cases}, \tag{4.2}$$

where μ is the friction coefficient, F_z is the vertical force due to its load, B and C are fit parameters, and the tire side slip angles of the rear and front tires are given by

$$\begin{aligned}
\alpha_F &= \tan^{-1}\left(\frac{v_y + L_f r}{v_x}\right) - \delta \\
\alpha_R &= \tan^{-1}\left(\frac{v_y - L_r r}{v_x}\right).
\end{aligned} \tag{4.3}$$

The tires cannot produce additional force after the slip angles pass a critical value α_{cr}. The lateral and longitudinal forces are bounded by the following circle

$$F_{yR}^2 + F_{xR}^2 \le (\mu F_z)^2, \quad F_{yF}^2 + F_{xF}^2 \le (\mu F_z)^2 \tag{4.4}$$

4.3 Hierarchical Route Guidance Framework adapted to the Scaled Platform

We have adapted the hierarchical framework (explained in Sec. 2) to our scaled platform as shown in Fig. 4.3. (x_r, y_r) is the reference position which the car is expected to reach from its current estimated position $(\hat{x}, \hat{y})$. For simplification and better understanding of the whole framework, we have divided it into 5 main blocks.

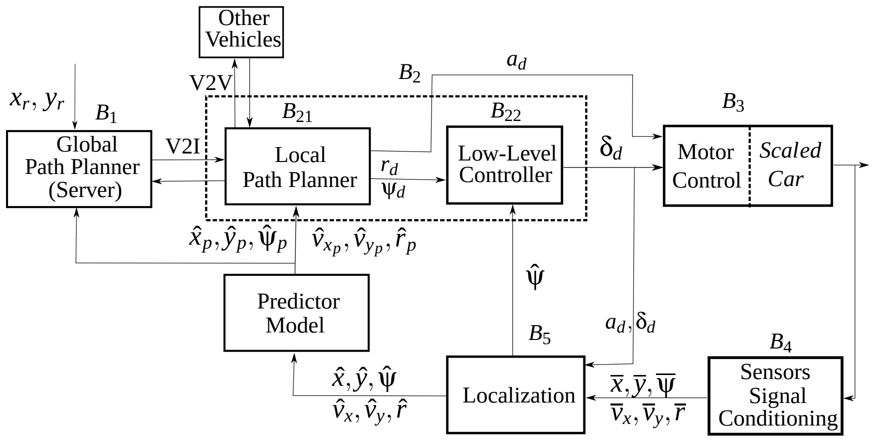

Other Vehicles
V2V
x_r, y_r
B_1
Global Path Planner (Server)
V2I
B_2
a_d
B_{21}
Local Path Planner
r_d
ψ_d
B_{22}
Low-Level Controller
δ_d
B_3
Motor Control
Scaled Car
$\hat{x}_p, \hat{y}_p, \hat{\psi}_p$
$\hat{v}_{x_p}, \hat{v}_{y_p}, \hat{r}_p$
$\hat{\psi}$
Predictor Model
a_d, δ_d
B_5
B_4
$\hat{x}, \hat{y}, \hat{\psi}$
$\hat{v}_x, \hat{v}_y, \hat{r}$
Localization
$\bar{x}, \bar{y}, \bar{\psi}$
$\bar{v}_x, \bar{v}_y, \bar{r}$
Sensors Signal Conditioning

Figure 4.3: Framework adapted for the current testing platform.

Block 1 (B_1) depicts the global path planner and V2X communication (includes both V2V and V2I). Block 2 (B_2) represents the local path planner and the low-level controller. Outputs of the local planner are the desired acceleration (a_d) and heading rate (r_d), while the low-level controller decides the steering angle demand (δ_d). Block 3 (B_3) details the motor control logic which involves the conversion of demand from B_2 to PWM signals. Block 4 (B_4) consists of the algorithms used for pre-processing of the sensor signals before being used for localization while Block 5 (B_5) consists of the localization algorithm. $(\bar{x}, \bar{y}, \overline{v_x}, \overline{v_y}, \overline{\psi}, \bar{r})$ represents the measured position, velocity, yaw angle and rate after sensor signal processing while $(\hat{x}, \hat{y}, \hat{v}_x, \hat{v}_y, \hat{\psi}, \hat{r})$ represents the estimated position, velocity, yaw angle and rate from the Kalman Filter. Block 6 (B_6) consists of a predictor model which helps to account for computational delay as well as latency of actuators. Blocks $B_1 - B_5$ are discussed in detail in Sections $4.4 - 4.8$. Blocks B_6 is discussed in detail in Section 5.3

4.4 Global Planner, B_1

A dynamic program (DP) is used for the global planner in order to obtain the cost-to-go approximation from various points on the terrain (defined on a grid) where the vehicle is supposed to travel. The choice of resolution or the size of each grid determines the computational complexity which increases exponentially with the increase in resolution. Hence, the global path planning is done using coarse resolution information. This cost map is then shared with the vehicles via V2I communication to guide their local planner. It has been discussed in detail in Sec. 3.1

4.5 Local Planner, B_2

A local path planner runs on-board the scaled car using Model Predictive Control (B_{21}). It is assisted by the cost map shared by the global planner and the information available via vehicle-to-vehicle (V2V) communication. One of the advantages of this process is maintaining a safe distance within the fleet without scattering. Also, using the cost map ensures the path tracked by the car is also globally optimal besides being locally optimal. A low-level controller is used for faster response due to the computational limitations of the on-board computer which make it difficult for the car to be directly controlled by the MPC. The MPC generates an optimal heading rate (r_d) and acceleration demand (a_d) which is used by the low level controller to steer the vehicle and apply the appropriate acceleration. (ψ_d) which is a by-product of the optimization and is used for robust steering control. MPC implementation is discussed in detail in Chapter 5.4.2

4.5.1 Low-Level Controller (PID)

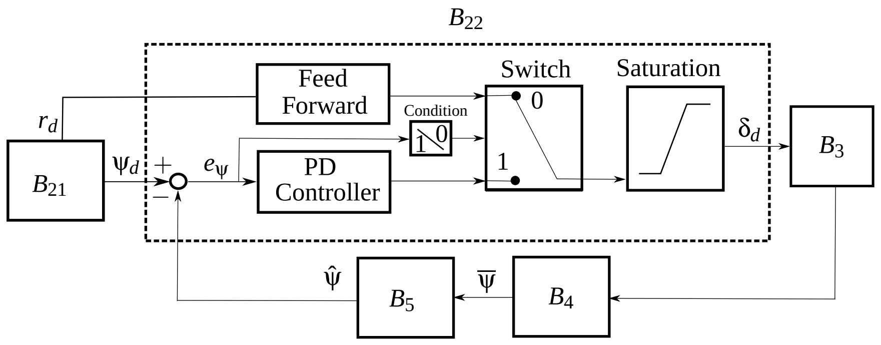


Figure 4.4: Block Diagram of the Low-Level Controller.

A schematic of the low level controller is shown in Fig. 4.4. A PD controller is used for controlling the steering which tracks the desired output of the local planner. Steering angle demand (δ_d) is the output of the low level controller which is then sent to the motor control block (B_3). $\hat{\psi}$ is the estimated heading which is one of the outputs of B_5 (explained in detail in Sec. 4.8). A feed-forward part is used when the error exceeds a set limit ($\pm e_\psi$). The feed - forward part uses the kinematic model of the vehicle to determine the desired steering angle given by Equation 4.5

$$\delta_d = \tan^{-1}\left[\frac{L_f}{L_f + L_r}\tan\left(\sin^{-1}\left(\frac{r_d L_r}{v}\right)\right)\right] \tag{4.5}$$

$$e_{\psi_{min}} \leq e_\psi \leq e_{\psi_{max}} \tag{4.6}$$

As a default, the switch is set to False, while it is set to True when the condition (4.6) is met. A saturation block is used to ensure that the demand steering is within limits.

4.6 Motor Control, B_3

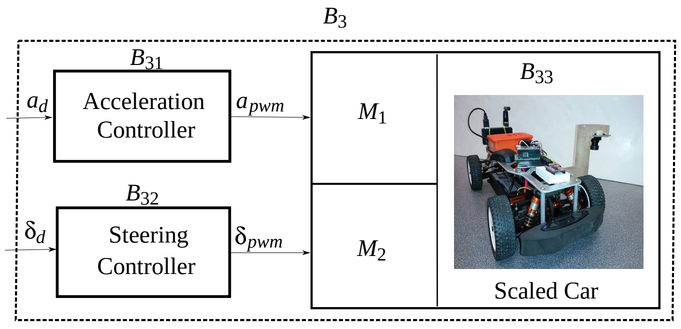


Figure 4.5: Overview of the motor control of the scaled car.

Motor Control Block (B_3) consists of a steering and acceleration controller. This block converts the mechanical demand (δ_d, a_d) to appropriate electrical signals (δ_{pwm}, a_{pwm}) for controlling the car. M_1 and M_2 represent the steering motor and the acceleration motor respectively.

4.6.1 Acceleration Controller

We have used a open-loop controller for the acceleration control. In Fig. 4.6 we can see the block diagram of the open-loop controller. The block B_2 sends a steering angle demand to the steering controller (B_{31}). The angle demand is then converted into appropriate PWM signal in B_{31}, which consists of a linear model (described by Equation 4.7).

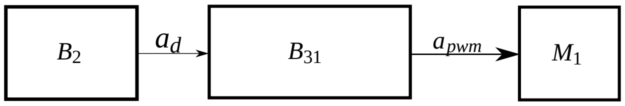


Figure 4.6: Block Diagram of the Open-Loop Acceleration Controller.

$$a_{pwm} = m_c + m_g a_d, \tag{4.7}$$

where m_c and m_g represents motor constant and motor gain respectively, which are calibrated based on the vehicle running ground truth data. While, a closed-loop acceleration controller was also developed but it lead to jerks while running the car. Hence currently, we have used the open loop controller for convenience.

4.6.2 Steering Controller

We have developed open-loop and closed-loop steering controllers. While the closed-loop control is more accurate, the advantage of the open-loop controller is the computational cost savings which we will discuss in detail in the next section.

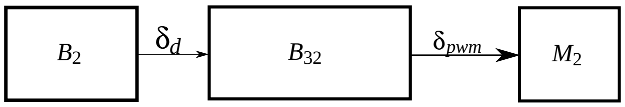


Figure 4.7: Block Diagram of the Open-Loop Steering Controller.

In Fig. 4.7 we can see the block diagram of the open-loop controller. The block B_2 sends a steering angle demand to the steering controller (B_{32}). The angle demand is then converted into appropriate PWM signal in B_{32}, which consists of polynomial models (described by Equation 4.8).

For model calibration, we made a sweep of the PWM signals and collected steering angle data as shown in Fig. 4.8. For steering angle measurement we are using a camera (explained in Sec. 4.6.3). From

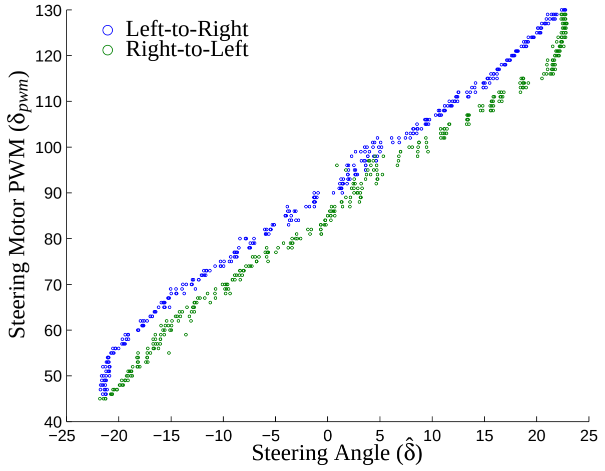


Figure 4.8: Plot of Observed Steering Angle ($\hat{\delta}$) vs Steering Motor PWM (δ_{pwm})

the data, we observed that the characteristic curve of the motor was dependent on the direction of turn of the wheels. The probable reason might be the hysteresis in the motor and mechanical linkages in the steering system. The angle (δ_d) to PWM (δ_{pwm}) conversion involves polynomial models as described in Equation 4.8.

$$\delta_{pwm} = c_1\delta_d^2 + c_2\delta_d + c_3 \tag{4.8}$$

where c_1, c_2, c_3 are the coefficients which are calibrated based on the experimental data as seen in Fig. 4.8. The coefficients are dynamically selected depending on the direction of rotation of the steering wheel, which takes into account the motor characteristics as observed in the experiments.

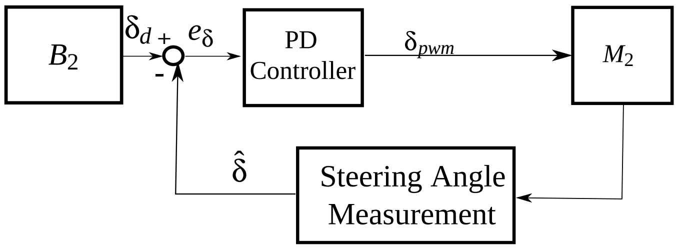


Figure 4.9: Block Diagram of Closed-Loop Steering Controller

A schematic of the closed-loop PD controller is shown in Fig. 4.9. The steering angle measurement is done in real-time using a camera mounted on to the chassis (refer to Fig. 4.1). e_δ is the steering angle error between the demand (δ_d) and observed steering angle ($\hat{\delta}$).

Dr. Hamed Saeidi contributed in development of the tire-edge detection using sobel operator [16].

4.6.3 Steering Angle Measurement

For the real-time measurement of steering angle $\hat{\delta}$, a USB camera is used with a tire edge detection algorithm. This algorithm uses the video frames from the top view of the wheel and detects the steering

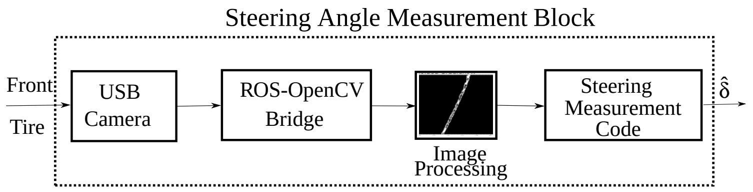


Figure 4.10: The internal block diagram of the Steering Angle Measurement block.

angle using a line fitted to the tire edge (see Fig. 4.11.d as an example). The slope of tire edge provides the steering angle $\hat{\delta}$ after a simple conversion according to what follows shortly. The internal block diagram of the Steering Angle Measurement in Fig. 4.9 is shown in Fig. 4.10. A built-in ROS-OpenCV bridge helps to obtain the video frames from the USB camera and process them in a ROS node using OpenCV library in order to detect the tire edge and extract its slope.

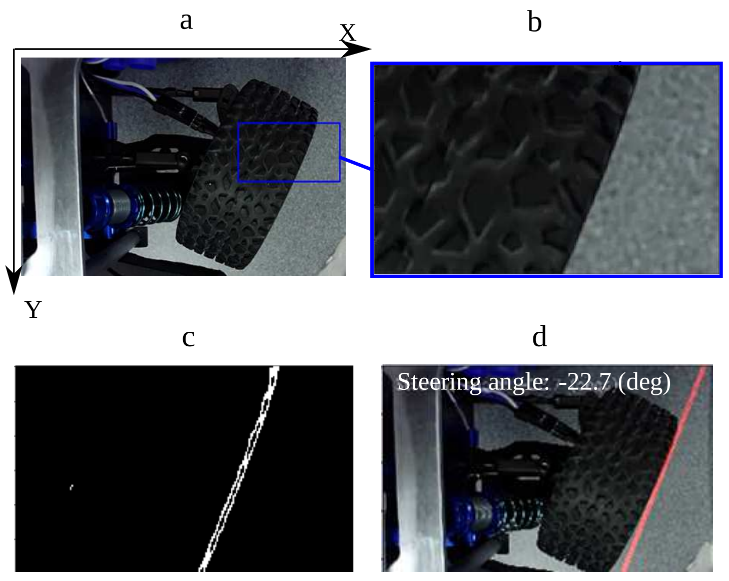


Figure 4.11: Steps of measuring steering angle $\hat{\delta}$ via the USB camera

These steps include, *a)* capturing a real-time video frame, *b)* cropping and scaling a useful portion of the image for edge detection, *c)* processing the results of step *b* and detecting the pixels corresponding to the tire edge, and *d)* fitting a line to the detected pixels and extracting the slope of line (i.e. $\hat{\delta}(t)$). For the last step, using the image frame coordinates shown in Fig. 4.11.a, we use the following equation to find $\hat{\delta}(t)$

$$\begin{aligned} x &= my + b \\ \hat{\delta}(t) &= \frac{180}{\pi} m, \end{aligned} \tag{4.9}$$

where m and b are the slope and offset of the line fit respectively.

4.6.4 Steering Angle Estimation

When the vehicle is running with the open-loop steering controller, we need an steering angle estimation algorithm which takes into account the system latency and response lag in the servo motor for improvement in localization.

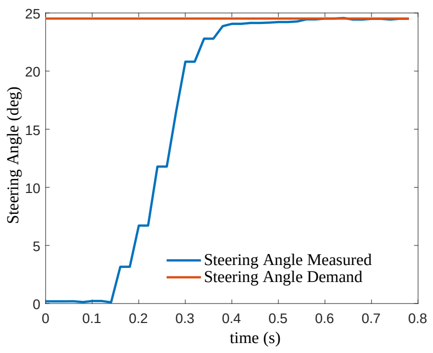


Figure 4.12: Response lag in the servo motor

In Fig. 4.12, we can see that the system latency is $\simeq 0.14$ s. We compensate this lag by using a ”double-ended queue” in Python, implementing First-In-First-Out (FIFO) data structure. The response lag is compensated by introducing a first order linear non-homogeneous differential equation (Equation 4.10).

$$\dot{\delta}_e(t) = -\frac{1}{\tau}\delta_e(t) + \frac{1}{\tau}\delta_d(t), \tag{4.10}$$

where δ_e and δ_d are the estimated and demand steering angle. The time constant τ is calibrated based on the experimental data. Figure 4.13 shows the response of the system defined by Equation 4.10 with respect to different values of τ.

Based on the optimum value of τ (we choose $\tau = 0.10$), the system written in discrete time for a sampling

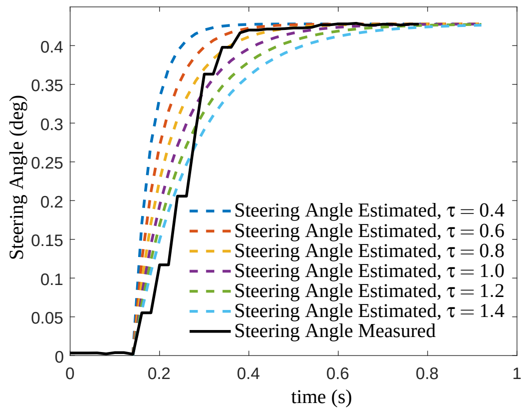


Figure 4.13: Steering model calibration for open-loop controller.

time of 0.02s is given by Equation 4.11

$$\delta_e(k+1) = \quad A\delta_e(k) + B\delta_d(k), \tag{4.11}$$

where A and B are the coefficients of the discrete time equation with values 0.6703 and 0.3297 respectively.

4.6.5 Comparison of the closed-loop and open-loop controllers

We performed experiments to compare the two controllers as shown in Fig. 4.14. Pre-defined steering angle inputs (δ_d) were given and the estimated steering angle ($\hat{\delta}$) was measured. It is observed that

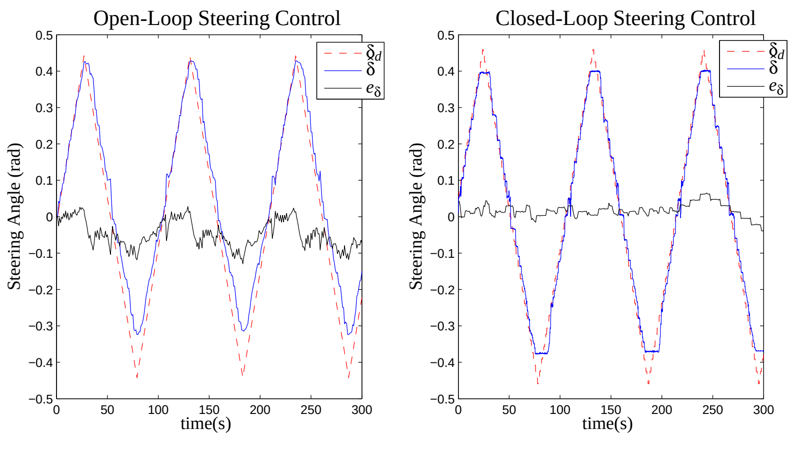


Figure 4.14: Comparison of open-loop and closed-loop control of steering.

the closed-loop controller is more effective in tracking the steering demand based on Fig. 4.14, where we can see that the tracking error (e_δ) is minimal. We tried to check the response of the open-loop system with random steering angle demand (δ_d) as shown in the Fig. 4.15. There is a clear lag in the demand and actual steering in the open-loop control which is mostly due to the latency and response lag. The closed-loop control takes care of this lag based on the measured steering angle feedback.

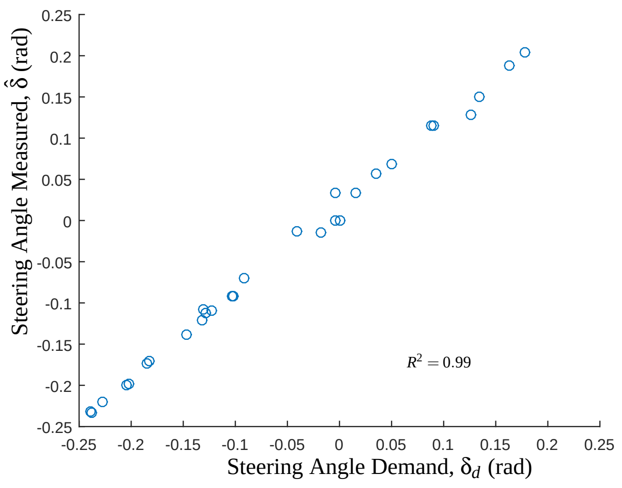


Figure 4.15: Response of the open-loop controller to random steering angle demand (δ_d).

The measurements were plotted taking into account the latency. The advantage of the open-loop controller is use of minimal computational resources with good tracking capability and response which makes it viable with the limited computational power in the on-board computer (2 GB RAM processor). In the future, we plan to use a more powerful on-board computer enabling the use of the closed-loop controller which can be vital when the vehicle is driven on off-road terrain.

4.7 Sensors Signal Conditioning, B_4

The raw signals from the sensors are processed before they are sent to the localization block, B_5 (refer Sec. 4.8). Encoder, IMU and GPS converter blocks are used for processing the signals from the wheel encoders, IMU and GPS respectively (Fig. 4.16).

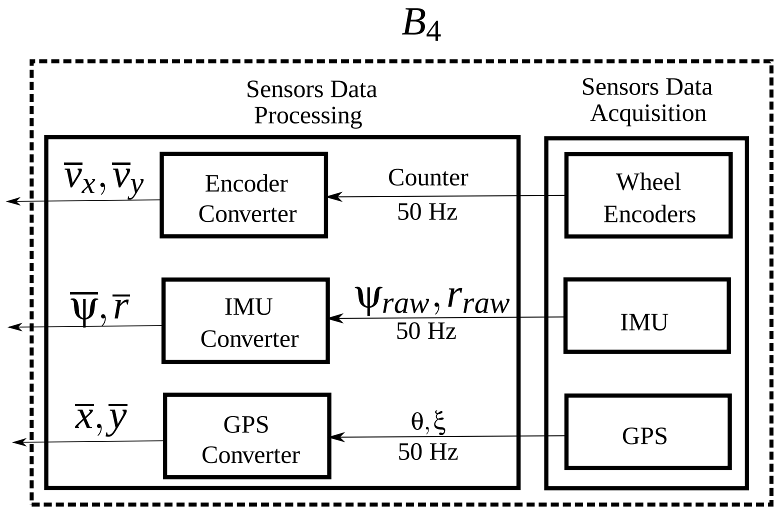


Figure 4.16: Sensor Signal Pre-Processing for Localization.

4.7.1 Encoder Converter

The encoder converter is used for vehicle speed measurement based on the raw signals from the hall effect sensors. Each wheel sends the encoder counter (C, which counts the number of times the magnets on the wheels have passed the sensor) from the Arduino, which is then used to measure the individual wheel speed as shown in Eq. (4.12).

$$v_i = \varkappa \frac{\Delta C_i}{\Delta t}, \quad i \in \{f_L, f_R, b_L, b_R\} \tag{4.12}$$

$$\varkappa = \frac{2\pi r_{tire}}{4} \tag{4.13}$$

where f_L, f_R, b_L, b_R corresponds to the front-left, front-right, back-left, back-right wheel. ΔC corresponds to the change in the wheel counter while Δt is the time between each measurement. r_{tire} is the radius of the tire and $\varkappa$ represents the distance along quarter tire edge as there are 4 equi-spaced magnets on the tire. Using simple kinematics, we derive the velocity of the car as shown in Equation. (4.14) based on v_{f_L}.

$$\begin{aligned} \overline{v}_x &= +\cos\hat{\psi}\left[v_{f_L}\cos(\hat{\delta}+\hat{\psi}) + \hat{r}d\sin(\lambda+\hat{\psi})\right] + \sin\psi\left[v_{f_L}\sin(\hat{\delta}+\hat{\psi}) - \hat{r}d\cos(\lambda+\hat{\psi})\right] \\ \overline{v}_y &= -\sin\hat{\psi}\left[v_{f_L}\cos(\hat{\delta}+\hat{\psi}) + \hat{r}d\sin(\lambda+\hat{\psi})\right] + \cos\psi\left[v_{f_L}\sin(\hat{\delta}+\hat{\psi}) - \hat{r}d\cos(\lambda+\hat{\psi})\right] \end{aligned} \tag{4.14}$$

where $\hat{\psi}, \hat{r}, \hat{\delta}$, are the estimated yaw angle, yaw rate and steering angle respectively. d represents the distance between the left-front wheel and the centre of mass of the vehicle while λ represents the angle made by front wheel hub with longitudinal axis of the car (X_b) (Fig. 4.2). The two components of vehicle velocity thus obtained are then sent to the Kalman Filter (explained in Sec. 4.8) as measurement signals after passing through a low-pass filter. Similarly, we can find $\overline{v}_x$ and $\overline{v}_y$ with respect to v_{f_R}.

4.7.2 IMU Converter

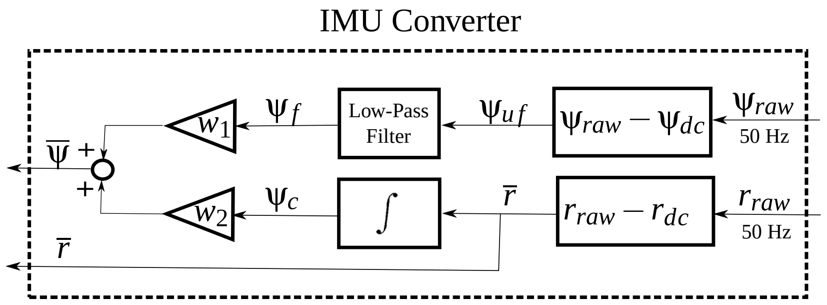


Figure 4.17: IMU Sensor Data Processing.

The IMU converter is used to process the raw yaw angle (ψ_{raw}) and raw yaw rate (r_{raw}) obtained from the IMU before sending it to the Kalman Filter as measurement signals. Fig. 4.17 shows the control flow of this algorithm. r_{dc} is a calibratable variable which is used to correct the sensor drift. To calibrate this variable, we measure the signals from the IMU in a standstill condition and study the drift in the sensor over time. In Fig. 4.18, we have a plot of the yaw angle (ψ_i) obtained by integrating the r_{raw} signals. The r_{raw} signals were analyzed and a glimpse of the same is shown in the zoomed view. We use the slope of a first order polynomial fit on the yaw angle data as a baseline for calibration of r_{dc}.

$$\psi_{dc}(k+1) = \psi_{dc}(k) - m(k)r_{dc}(k) \tag{4.15}$$

ψ_{dc} is defined by Eq. 4.15 where the counter (m) is incremented every time a signal passes.

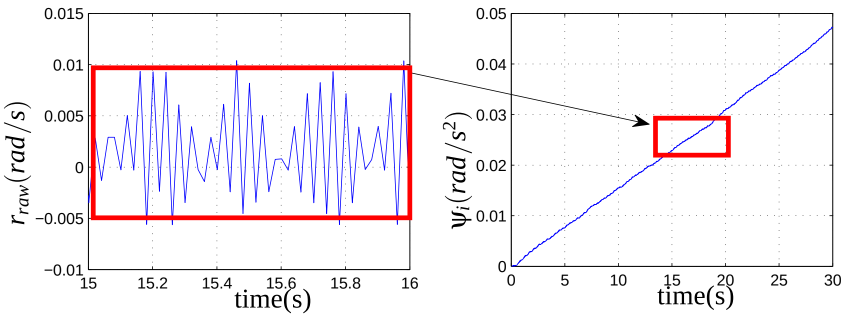


Figure 4.18: Drift in IMU sensor measurement

The r_{dc} takes into account the influences of magnets used for speed measurement as well as the electric field generated by the motor. Hence, fitting the exact slope might not work for accurate estimation and we need to calibrate it based on ground truth data. Two weighing variables w_1 and w_2 are used to fuse the filtered yaw angle (ψ_f) and corrected yaw angle (ψ_c) based on the ground truth data to get the processed yaw angle ($\overline{\psi}$).

4.7.3 GPS Converter

A GPS converter is used to convert the latitude/longitude values from the GPS sensors into the Cartesian coordinate system according to the equation 4.16.

$$\begin{aligned} \overline{x} &= R(\xi - \xi_0)\cos\theta \\ \overline{y} &= R(\theta - \theta_0) \end{aligned} \tag{4.16}$$

where R, ξ, θ corresponds to the radius of the earth, instantaneous longitude and latitude respectively. ξ_0, θ_0 corresponds to the latitude and longitude of the initial starting point of the car. This is a simplified form arrived at using geometry with the assumption that the robot travels a small distance ($\simeq$ 200m). In the future, we plan to use the haversine formula which take into account the curvature of the earth.

4.8 Localization, B_5

We use an Extended Kalman Filter for localization of the vehicle as shown in Fig. 4.19. The measurements from the localization sensors are pre-processed as explained in Sec. 4.7. The control input is received from the output of the Low-Level Controller.

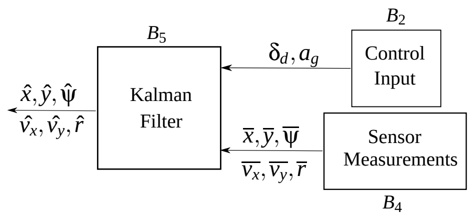

B_5
Kalman Filter
B_2
Control Input
δ_d, a_g
Sensor Measurements
B_4
$\overline{x}, \overline{y}, \overline{\psi}$
$\overline{v_x}, \overline{v_y}, \overline{r}$
$\hat{x}, \hat{y}, \hat{\psi}$
$\hat{v}_x, \hat{v}_y, \hat{r}$

Figure 4.19: Overview for Localization of the scaled car.

Denote $\mathbf{x} = [x, y, \theta, v]$ as the state vector, $\mathbf{u} = [\delta, a]$ as the input vector. Therefore, using the standard EKF notation, we define $\mathbf{x}_{k|k-1}$ and $\mathbf{x}_{k|k}$ as the *a priori* and *a posteriori* estimated state at the time step k, $\mathbf{x}_{k-1|k-1}$ as the most recent updated state before the current step k. Similar notation applies to the state covariance matrix $S \in \mathbb{R}^{4\times4}$. The non-linear dynamic equations of motion $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \nu_k)$, using a first-order Euler approximation, are given as follows

$$\begin{bmatrix} X_{k+1} = X_k + (v_{x_k}\cos\psi_k - v_{y_k}\sin\psi_k)\Delta t \\ Y_{k+1} = Y_k + (v_{x_k}\sin\psi_k + v_{y_k}\cos\psi_k)\Delta t \\ \psi_{k+1} = \psi_k + r\Delta t \\ v_{x_{k+1}} = v_{x_k} + (\frac{F_{xRk} - F_{yF\,k}\sin\delta_k}{m} + v_{y_k} r_k)\Delta t + \nu_1 \\ v_{y_{k+1}} = v_{y_k} + (\frac{F_{yR_k} + F_{yF\,k}\cos\delta_k}{m} + v_{xk} r_k)\Delta t \\ r_{k+1} = r_k + (\frac{L_f F_{yF\,k}\cos\delta_k - L_r F_{yR_k}}{I_z})\Delta t + \nu_2 \end{bmatrix} \tag{4.17}$$

where $\nu_k = [\nu_1, \nu_2]$ is the process noise and $\nu_1 = \mathcal{N}(0, \sigma^2_{F_{xR}})$ and $\nu_2 = \mathcal{N}(0, \sigma^2_{\delta})$ are random samples representing unknown effects of traction force and steering angle. Denote the measurement model as $\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{n}_k)$ where $\mathbf{n}_k$ is the measurement/observation noise. When using the IMU and wheel encoder sensors, $\mathbf{h}(\mathbf{x}_k, \mathbf{n}_k)$ takes the following form,

$$\mathbf{h}_1(\mathbf{x}_k, \mathbf{n}_1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{n}_1, \tag{4.18}$$

and when using the GPS sensor measurements it takes the following form

$$\mathbf{h}_2(\mathbf{x}_k, \mathbf{n}_2) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{n}_2, \tag{4.19}$$

where $\mathbf{n}_1 = [n_{\text{IMU}}, n_{\text{enc}}]$ and $\mathbf{n}_2 = [n_{\text{GPSx}}, n_{\text{GPSy}}, n_{\text{IMU}}, n_{\text{enc}}]$ are the observation noise via different sensors. For these noises, we assume a Gaussian distribution $n_{\text{IMU}} = \mathcal{N}(0, \sigma_{\text{IMU}})$, $n_{\text{enc}} = \mathcal{N}(0, \sigma_{\text{enc}})$, $n_{\text{GPSx}} = \mathcal{N}(0, \sigma_{\text{GPSx}})$, and $n_{\text{GPSy}} = \mathcal{N}(0, \sigma_{\text{GPSy}})$ where σ_{IMU}, σ_{enc}, σ_{GPSx}, and σ_{GPSy} are the standard deviations. The equations for the EKF prediction step are according to the following

$$\begin{aligned} \mathbf{x}_{k|k-1} &= \mathbf{f}(\mathbf{x}_{k-1|k-1}, \mathbf{u}_{k-1}) \\ S_{k|k-1} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}} S_{k-1|k-1} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{\mathrm{T}} + \frac{\partial \mathbf{f}}{\partial \nu} Q \frac{\partial \mathbf{f}}{\partial \nu}^{\mathrm{T}}, \end{aligned} \tag{4.20}$$

and for the update step are according to the following

$$\begin{aligned} \mathbf{K}_k &= S_{k|k-1} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\mathrm{T}} \left[\frac{\partial \mathbf{h}}{\partial \mathbf{x}} S_{k|k-1} \frac{\partial \mathbf{h}}{\partial \mathbf{x}}^{\mathrm{T}} + \frac{\partial \mathbf{h}}{\partial \mathbf{n}} R \frac{\partial \mathbf{h}}{\partial \mathbf{n}}^{\mathrm{T}} \right]^{-1} \\ \mathbf{x}_{k|k} &= \mathbf{x}_{k|k-1} + \mathbf{K}_k [\mathbf{y}_k - \mathbf{h}(\mathbf{x}_{k|k-1})] \end{aligned} \tag{4.21}$$

$$S_{k|k} = \left[I - \mathbf{K}_t \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right] S_{k|k-1}. \tag{4.22}$$

In Equation. (4.21), $\mathbf{K}_k$ is the Kalman gain. Here, assuming independence of the measured variables, we have $Q = \text{diag}[0, 0, \sigma_{F_{xR}}, \sigma_\delta]$ as the dynamic noise covariance matrix, and $R_1 = \text{diag}[\sigma_{\text{IMU}}, \sigma_{\text{enc}}]$ and $R_2 = \text{diag}[\sigma_{\text{GPSx}}, \sigma_{\text{GPSy}}, \sigma_{\text{IMU}}, \sigma_{\text{enc}}]$ as the measurement noise covariance matrices.

It can be easily shown that the position state vector is not observable using only IMU and wheel encoder sensors. Moreover, ψ is not observable by using only GPS and wheel encoder sensors. Here, we fuse these sensors with the algorithm shown in Algorithm 1 to obtain a more precise tracking of the location of the vehicle. In the fusion algorithm, each time a new measurement from the sensors is obtained the state estimations are updated using the observation models (i.e. $\mathbf{h}_1$ and $\mathbf{h}_2$) corresponding to the received sensory data. The details are of calculating the $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, $\frac{\partial \mathbf{f}}{\partial \nu}$, $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$, and $\frac{\partial \mathbf{h}}{\partial \mathbf{n}}$ are trivial and hence omitted here.

Algorithm 1 The sensor fusion algorithm for localization.

```
1: procedure FUSING THE IMU, WHEEL ENCODER, AND GPS MEASUREMENTS TO ESTIMATE $\mathbf{x}$
2:     while Experiment is running do
3:         Calculate $\Delta t$
4:         Implement EKF in (4.20) and (4.21) via
5:         if GPS measurement unavailable then
6:             $\mathbf{h}(\mathbf{x}) = \mathbf{h}_1(\mathbf{x})$ and $R = R_1$.
7:         else if GPS measurement available then
8:             $\mathbf{h}(\mathbf{x}) = \mathbf{h}_2(\mathbf{x})$ and $R = R_2$.
9:         end if
10:    end while
11: end procedure
```

4.9 Experiments: Localization and Low-Level Controller

To analyze the efficacy of our localization and low-level control algorithms, we bypassed the global and the local path planner by providing pre-defined inputs to the low-level controller (ψ_d, a_d) and observed the behaviour of the steering control (δ_{pwm}) and the accuracy of position estimation ($\hat{x}, \hat{y}$) and yaw angle ($\hat{\psi}$) with ground truth data. After a pre-defined time, the control action is set to zero to bring the car to a standstill. After the localization and control algorithm were verified, we used the local planner to test the path planning as well as the obstacle avoidance algorithm's effectiveness.

4.9.1 Localization Accuracy

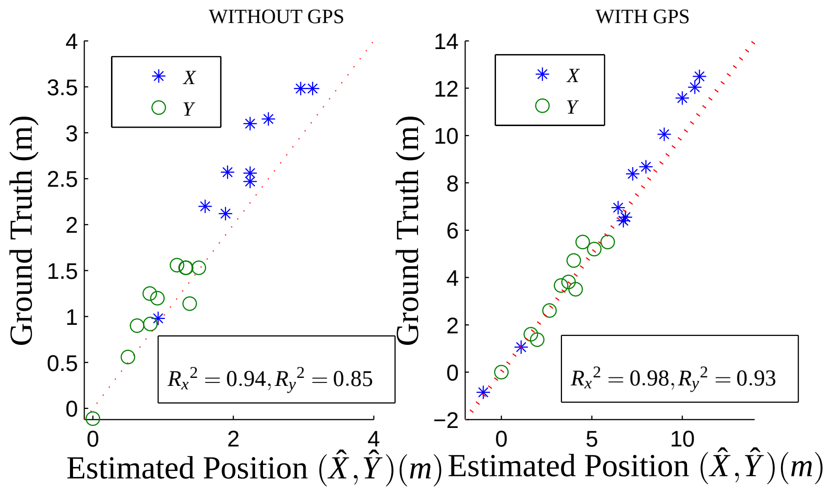


Figure 4.20: Position Estimation with and without GPS

The vehicle was run for varying inputs (ψ_d, a_d) and it was found that for up to 4 seconds, the estimations were acceptable (error within %) when GPS signal was unavailable. Without GPS, the system is inherently unobservable, and hence any error in measurement leads to the states growing unbounded. This analysis is important for finding the limitations of accurate estimations in case of GPS signal drop or missing. A low-cost GPS which normally runs at a frequency of 1 Hz can be used to perform relatively good localization using our algorithms. We measured the ground truth data at the end of each test using an inclinometer and a rolling measuring wheel and compared with the estimated position ($\hat{x}, \hat{y}$) as well as yaw angle ($\hat{\psi}$) of the car. Our future plan is to use an accurate high frequency positioning system to measure the accuracy of the estimations at each time instant.

Fig. 4.20 shows the ground truth data against the estimated position ($\hat{x}, \hat{y}$) in both cases, with and without the GPS signal available. With GPS, the system is observable, and hence it is predicted to improve accuracy of the state estimation. The estimations improved as predicted, which can be judged by comparison of the R^2 value in both cases.

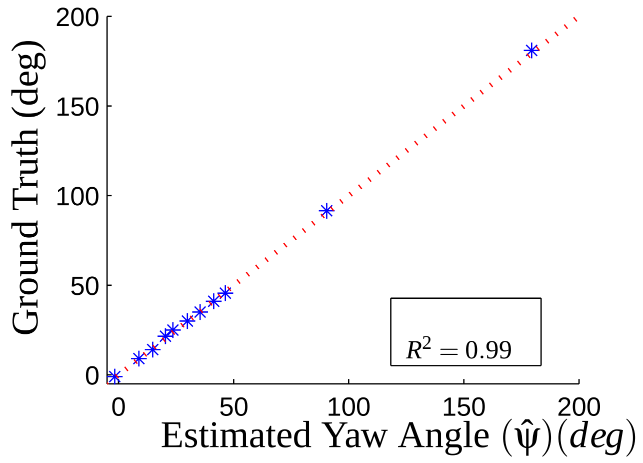


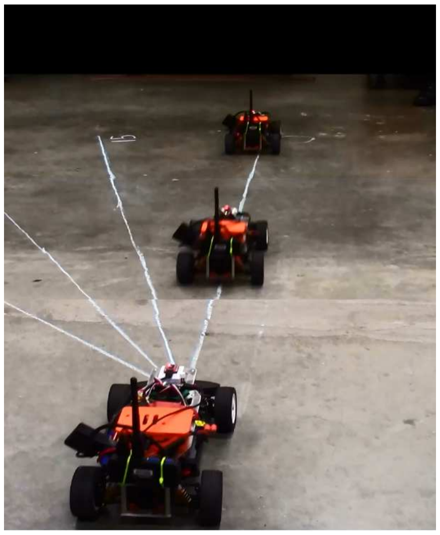
Figure 4.21: Yaw Angle Estimation

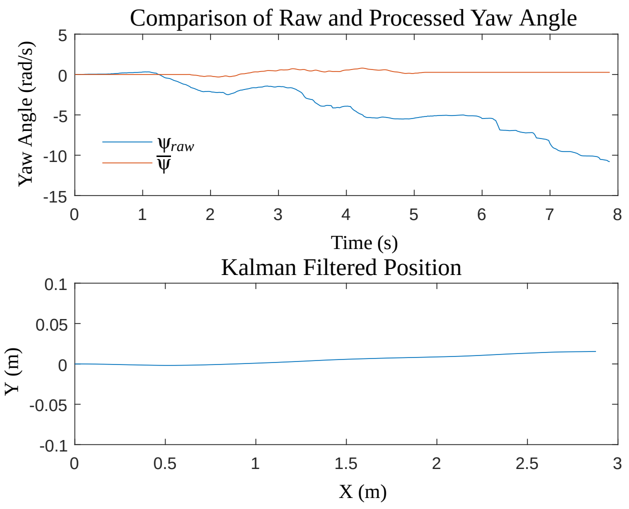
Fig. 6.1 shows the predicted yaw angle against the ground truth data. This relatively accurate estimations were possible due to the drift correction algorithm in the IMU converter as shown in Fig. 4.22. It can been seen that the there is a clear drift in the raw yaw angle (ψ_{raw}) primarily arising due to the raw yaw rate r_{raw} as shown in Fig. 4.18. The high accuracy can also be attributed to the fact that the car was driven for $\leq$15 m. When it is run for longer durations or length, the drift will eventually creep in, but our target was to improve it in the local domain so that a low frequency signal can be used for any corrections.

Fig. 4.22 which shows the movement of the vehicle with $\psi_d = 0$ and underlines the importance of the IMU converter where the raw signals are processed for drift correction to accurately estimate the yaw angle.

4.9.2 Low-Level Controller

We ran multiple tests to analyze the efficacy of the control algorithm of the low-level controller in conjunction with the localization algorithm. Fig. 4.22 is one of the tests where we used $\psi_d = 0$. In the discussions below, we have a few tests where we used ψ_d = $\{10°, 45°, 90°\}$, 180° with fixed a_d for collecting data and analysis. The role of the low-level controller is to take the vehicle to the demand heading (δ_d). During the tests the feed-forward part of the controller which utilizes the output (r_d)of the MPC, is bypassed and it runs purely as a PD controller which is ensured by changing the de-activation of the Feed-Forward part


(a) Scaled Car running with $\psi_d = 0$



(b) Data captured during the experiment

Figure 4.22: Advantage of the processing the IMU signal in the IMU converter

to TRUE.

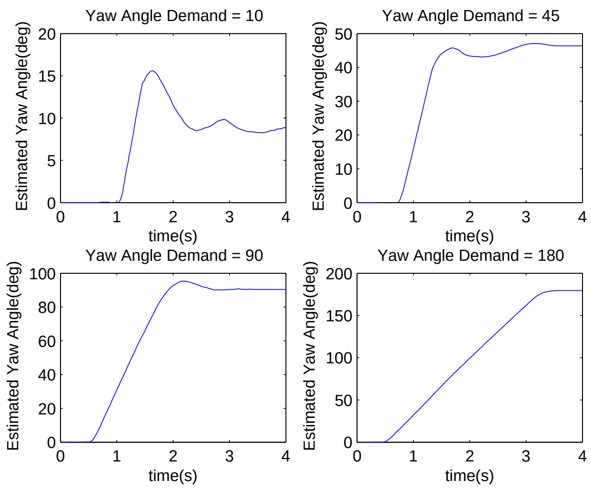


Figure 4.23: Yaw Control.

In Fig. 4.23, we can see the estimated yaw angle ($\hat{\psi}$) as the vehicle moves to achieve the demand yaw angle (ψ_d). The overshoot at $\psi_d = 10^\circ$ was due to the higher PD gain which had to be applied so that the handling is not too sluggish with higher ψ_d. The estimated yaw angle was compared to the ground truth at the end of each test and they were within $\pm 1\%$. We also run multiple tests with different PD gains and found a need for using a gain scheduler or a phase lead compensator to improve vehicle handling for stability at wide ranges of ψ_d. We will see in Sec. 5.4 how implementation of the feed-forward part in the controller in a way acts as compensator or variable gain controller.

The development of the scaled car along with accurate localization and low-level had been discussed in our research which is under review [16].

Chapter 5

MPC-based Obstacle Avoidance

5.1 MPC Formulation

The local route guidance routine is based on a model predictive control approach. The optimal local route in the immediate neighbourhood of a vehicle is calculated in a receding horizon manner as the vehicle navigates through the terrain. We propose to use a simple kinematic model of the vehicle [19] over the MPC horizon with 4 states: X and Y are the coordinates of the center of gravity with respect to vehicle's initial position and ψ is the vehicle heading angle with respect to the fixed coordinate frame, $X_a - Y_a$ (refer Sec. 4.2) and v is the vehicle velocity. The input to this model is the commanded change in steering angle (δ_d) and the acceleration (a) . The dynamic equations of the vehicle model are shown in Equation 5.1

$$\begin{aligned}
\dot{X} &= v\cos\psi \\
\dot{Y} &= v\sin\psi \\
\dot{\psi} &= \frac{v}{L_b}\sin\beta \\
\dot{v} &= a_d \\
\beta &= \tan^{-1}\left(\frac{L_f}{L_f + L_r}\tan\delta_d\right),
\end{aligned} \tag{5.1}$$

where L_f and L_r represent the distance of front and rear wheel axles from the CoG of the car.

$$\begin{aligned}
\dot{X} &= v\cos\psi \\
\dot{Y} &= v\sin\psi \\
\dot{\psi} &= r_d \\
\dot{v} &= a_d,
\end{aligned} \tag{5.2}$$

Another MPC model as shown above was developed where the yaw rate (r_d) is used as the control input instead of steering angle. This provided us more control on the movement of the car as we can constrain the rate of turning of the vehicle. We had conducted two sets of experiments: 1) with the vehicle directly controlled by the MPC (using Equation 5.1) which runs at 2 Hz and, 2) the low level controller running at 10 Hz directly controlling the vehicle with inputs from the MPC (using Equation 5.2) at 2 Hz. We saw that with the second experiment vehicle control was much more smoother since in the first experiment, the MPC can correct for any modeling error every 0.5 seconds while the low-level can correct every 0.1 seconds. Also to have more control on the rate of turning of the vehicle, in the kinematic model defined by Equation 5.1 we would have to introduce steering angle as a new state and steering rate as the control input which constraints the computational resources. We have also developed a simulation environment where the plant is based on the vehicle kinetic model (Section 4.2). Our simulation results also showed the need of using a lower level controller (Section 4.5.1) for faster control of the steering, rather than MPC controlling steering command. Hence we have proceeded using equation 5.3 for the rest of our research. Using Euler's Method, the discretized form of the MPC is given by

$$\begin{aligned}
X(k+1) &= X(k) + \Delta t\left[v(k)\cos\psi(k)\right] \\
Y(k+1) &= Y(k) + \Delta t\left[v(k)\sin\psi(k)\right] \\
\psi(k+1) &= \psi(k) + \Delta t\left[r_d(k)\right] \\
v(k+1) &= v(k) + \Delta t\left[a_d(k)\right],
\end{aligned} \tag{5.3}$$

The objective of the modified MPC optimization routine is to,

$$\begin{aligned}\min_{r_{d_1}\cdots r_{d_n}, a_{d_1}\cdots a_{d_n}} \quad J = w_1 \sum_{k=1}^{n} \Delta l_k + w_2 \sum_{k=1}^{n} {r_{d_k}}^2 + w_3 \sum_{k=1}^{n} {a_{d_k}}^2 + w_4 \sum_{k=1}^{n} {\varepsilon_k}^2 \\ w_5 \sum_{k=1}^{n} vis_k + w_6 \sum_{k=1}^{n} soil_k + J^*_{DP}(x_n, y_n)\end{aligned} \tag{5.4}$$

with the following inequality constraints,

$$\begin{aligned} r_{min} &\leqslant r_k \leqslant r_{max} \\ a_{d_{min}} &\leqslant a_{d_k} \leqslant a_{d_{max}} \\ v_{min} &\leqslant v_k \leqslant v_{max} \\ 0 &\leqslant \varepsilon_k \\ d_{safe} &\leqslant d_{obs_k} + \varepsilon_k \end{aligned} \tag{5.5}$$

In the cost function, n is the prediction horizon and, r_k and a_k are the free optimization variables at successive steps; each represents the commanded change in heading angle and velocity at stage i. The penalty weights are design variables and denoted by w_i. The seven terms of the cost functions are respectively:

- Δl_k is the 2D distance covered in the k^{th} step which is computed as,

$$\Delta l_k = (x_r - x_k)^2 + (y_r - y_k)^2 \tag{5.6}$$

- To ensure that minimal control effort is used in achieving the objective of the optimization routine, we penalize the control inputs (a_{d_k}, r_{d_k}).
- To ensure that the local route prescribed avoids collision with the obstacles we penalize the locality of the detected obstacles heavily. For an obstacle q, let (x_{obs_q}, y_{obs_q}) denote the geometric center of the obstacle q, while d_{safe} denotes the preferred safe distance between the vehicle and obstacle. Introduction of ε_k in Equation 5.5 softens the obstacle avoidance constraint. The penalty weight on the obstacle avoidance cost is set to zero if the obstacle is outside the unsafe zone of the vehicle which is determined using a barycentric method (explained in detail in Sec 5.2).

- To reduce the chance of being detected, the cost function (5.4) penalizes areas which can be seen from the adversarial towers by introducing the cost term,

$$visibility_k = \begin{cases} 0 \quad \text{if} \quad x_k, y_k \in invisible, \\ \\ 1 \quad \text{if} \quad x_k, y_k \in visible. \end{cases} \tag{5.7}$$

- The function $soil_k$ is the computed value associated with the terrain traversed as described in section 2.2. Hence the cost functions ensure that the prescribed route utilizes the best soil possible.

- The last term $J^*_{DP}(x_n, y_n)$ represents the optimal cost-to-go, calculated by the dynamic program (Sec. 4.4), from the end of the MPC horizon to the final target. Inclusion of the cost-to-go is intended to prevent short-sighted decisions by the local path planner and ensures local decisions are guided by global objectives.

The cost function in (5.4) is minimized subject to the vehicle kinematic equations in (5.3) and the constraints in (5.5). This problem is solved numerically with an interior point method using Julia which is a high-level, high-performance programming language for numerical computing. For implementation of the interior point method, we use the IPOPT (Interior Point OPTimizer) software package. The cost function associated with soil, visibility and optimal cost-to-go has been validated in our preliminary simulation studies which showed how they can improve the path planning (Sec 3.3). In the context of this thesis, we will not use them in our experiments due to the computational limitations associated with running the interpolation algorithm to implement in the MPC, in real-time.

A subject of our ongoing research is to use a higher fidelity vehicle model to evaluate the performance of the vehicle when following the MPC command, which is beyond the scope of this thesis. We had to restrict ourselves to a simpler model due to the computational limitations of running MPC in the on-board computer. In our future research, we plan to use the dynamic model (refer Sec. 4.2) to run the MPC with more powerful on-board computer.

5.2 Obstacles, Safe and Unsafe Zones

In this thesis, we have used virtual obstacles for our experiments in place of real obstacles since we wanted to concentrate on developing the hierarchical platform as well as high and low-level control of the car. Using a LIDAR and stereo camera for obstacle detection was out of the scope of this research. Two types of obstacles have been created, i) whose location are known a-priori ii) which are recognized based on the simulated sensor horizon.

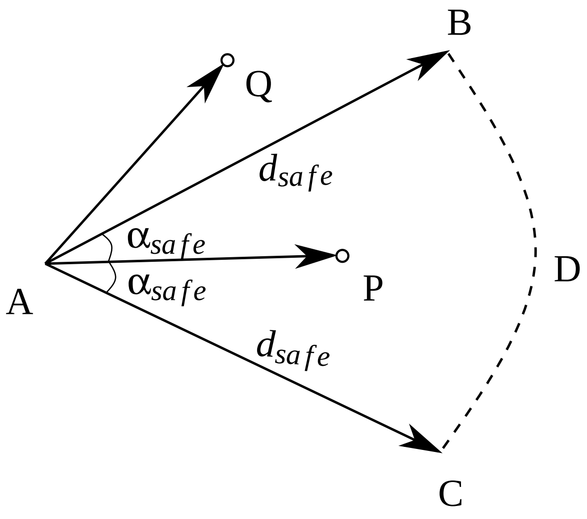


Figure 5.1: Barycentric co-ordinate approach in determining the safe and unsafe zones for the vehicle.

d_{safe} and α_{safe} are the preferred safe distance and angle. They are used as constraints for defining the safety region for the vehicle to traverse. The region ABDC as shown in Fig. 5.1 is considered as unsafe zone, where A is the position of the vehicle and, P and Q are the position of two obstacles. The region ABDC is shown as an unsafe zone so ensure the vehicle takes course correction if any obstacle is seen in that area, while obstacles outsize the zone, it doesn't need to take immediate course correction and can continue on its planned route. To determine if a obstacle is in the safe or unsafe zone of the vehicle, we used a obstacle detection algorithm based on the work by Ericson [12]. For an obstacle q, let A(q) denote the unsafe region. Any point, P on the plane can be written as,

$$\overrightarrow{AP} = u\overrightarrow{AB} + v\overrightarrow{AC}, \tag{5.8}$$

where, u and v are two arbitrary constants which can be calculated if the position of the obstacle is known. Using simple vector algebra, we can find, u and v, as shown in Equation 5.9

$$
\begin{aligned}
u &= \frac{(\overrightarrow{AB}.\overrightarrow{AB})(\overrightarrow{AP}.\overrightarrow{AC}) - (\overrightarrow{AB}.\overrightarrow{AC})(\overrightarrow{AP}.\overrightarrow{AB})}{(\overrightarrow{AC}.\overrightarrow{AC})(\overrightarrow{AB}.\overrightarrow{AB}) - (\overrightarrow{AC}.\overrightarrow{AB})(\overrightarrow{AB}.\overrightarrow{AC})} \\
v &= \frac{(\overrightarrow{AC}.\overrightarrow{AC})(\overrightarrow{AP}.\overrightarrow{AB}) - (\overrightarrow{AC}.\overrightarrow{AB})(\overrightarrow{AP}.\overrightarrow{AC})}{(\overrightarrow{AC}.\overrightarrow{AC})(\overrightarrow{AB}.\overrightarrow{AB}) - (\overrightarrow{AC}.\overrightarrow{AB})(\overrightarrow{AB}.\overrightarrow{AC})}
\end{aligned}
\tag{5.9}
$$

So the condition necessary to be satisfied for the obstacles to be in the unsafe zone is given by equation 5.10.

$$
\begin{cases}
P \notin \bigcup_{q=1}^{Q} A(q) & \text{if} \quad u^2 + v^2 > 1, u < 0, v < 0 \\
P \in \bigcup_{q=1}^{Q} A(q) & \text{if} \quad u^2 + v^2 \leq 1, u \geq 0, v \geq 0
\end{cases}
\tag{5.10}
$$

Hence for any obstacle detected in the region will activate the penalty function in the MPC cost, J, while for any other obstacles detected outside of this zone, penalty weight will be set to zero. The penalty weight w_4 in the cost function (5.4) is defined as,

$$
w_4 = \begin{cases}
0 & \text{if} \quad P \notin \bigcup_{q=1}^{Q} A(q), \\
10^5 & \text{if} \quad P \in \bigcup_{q=1}^{Q} A(q).
\end{cases}
\tag{5.11}
$$

The obstacle detection algorithm runs in Python and provides the MPC running Julia with obstacle information. For reducing the computational cost in obstacle detection, we have used a method of priority queuing in Python, which considers the closed two obstacles in the unsafe zone.

5.3 Computational Delay Compensation and Predictor Model, B_6

Predictive control involves on-line optimization which might lead to considerable computational delay if we use an underpowered computing resource. The Fig. 5.2, taken from Machiejowski's book "Predictive Control with Constraints" [27] helps in explaining the need to take into account the computational delay in predictive control.

In our case, it was an important to factor in this delay as we were running a lot of computations besides the MPC in a low computationally powered computer (2 GB RAM). We also took into consideration the latency for every control input update. So the MPC state estimates are predicted for the time when control

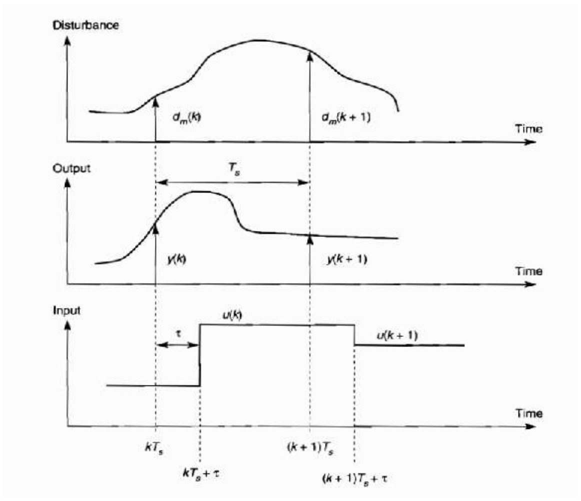


Figure 5.2: Assumed timing of measuring and applying signals (Figure adopted from [27]).

input is actually applied by the actuators. We have used the concept of constant output disturbance observer in this predictor model to take into account the model mismatch in prediction. As shown in equation 5.12, the disturbance/model mismatch is calculated by comparing the measured output with the predicted one. It is then added to the predicted output for the next state. Here we consider that the control inputs don't change during the prediction interval.

$$\hat{d}(k|k) = y_p(k) - \hat{y}_p(k|k-1) \tag{5.12}$$

$$\hat{y}_{p_{corr}}(k+1|k) = \hat{y}_p(k+1|k) + \hat{d}(k|k) \tag{5.13}$$

5.4 Experiments: Local Planner and Obstacle Avoidance

In the following experiments, we define the reference position (x_r, y_r) as the target position for the vehicle to reach. We conducted experiments with and without obstacles in its path to analyze the efficacy of the MPC in optimally avoiding the obstacles as well as reach its target position.

5.4.1 Local Planner

As a part of testing the local planner, we initially tested with different reference target position (x_r, y_r). Below we detail the results of one set of experiments where (x_r, y_r) was (3,2.5). During our experiments with the local controller we found that it will be useful to have a gain scheduler or a compensator for better vehicle control. We ran three sets of experiments with the same reference target and analyzed the outputs: 1) only PD controller activated by setting the switch condition to 1 throughout the experiment, 2) only Feed-Forward part activated by setting the switch condition to 0 throughout the experiment, 3) the activation of the PD controller and the Feed-Forward part based on switch condition given by Equation 4.6 throughout the experiment. For better understanding of the three experiments please refer to Figure 4.4. Hence test 1 was running only on the PD controller, test 2 was running only on the feed-forward part and test 3 was running on a optimal combination of the two. The input to both the PD and the feed-forward part came from the MPC controller and hence this test also helped us to analyze how the MPC in conjunction with the low-level controller was performing as the local planner.

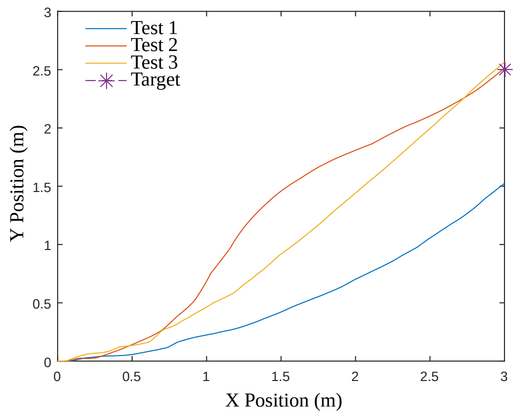


Figure 5.3: Vehicle path trace running MPC along with different settings in the low-level controller

We can see from Fig 5.3, in test 1, the steering controller is too slow while in test 2 it is very aggressive. We can tune the PD gain to improve the results in test 1, but then we will have overshooting issues as in Sec. 4.9.2. Hence an optimal combination of the two is useful.

From Fig. 5.4 we can see that the combined controller is more efficient since it uses a higher gain which corresponds to bigger steering angle demand when the e_ψ which is the difference between ψ_d and $\overline{\psi}$ is high while it uses much smaller steering demand when e_ψ is small.

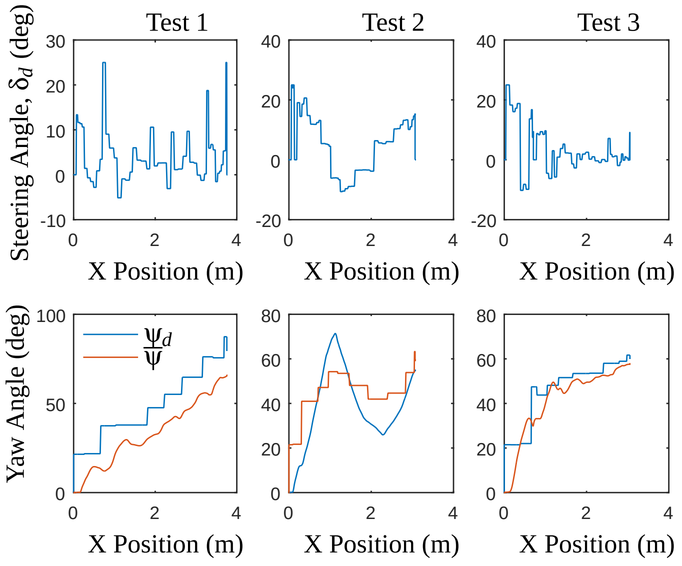


Figure 5.4: Trace of Steering Angle and Yaw Angle in the three tests

5.4.2 Obstacle Avoidance: Stationary and Moving

We ran a few tests to test the obstacle avoidance capability of the MPC with virtual obstacles. To compare how the MPC performs the local planning with varying constraints, we ran two tests on the vehicle with the same reference target position (x_r, y_r) which was (10,10).

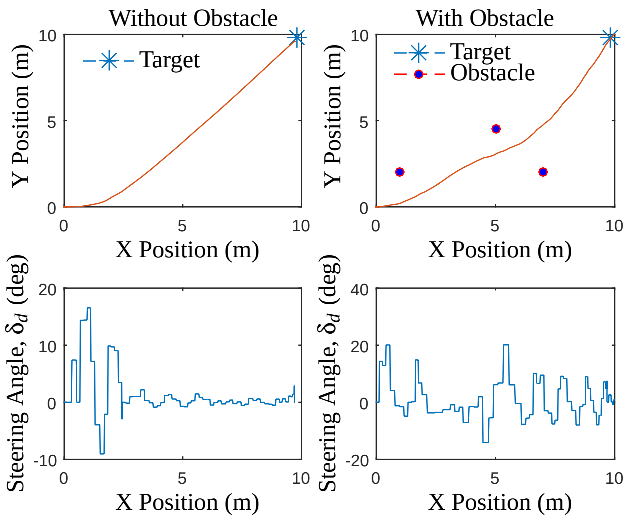


Figure 5.5: Vehicle path trace and steering command in with and without obstacle scenarios.

In the test 1, there were no obstacles while in test 2 we had a few virtual obstacles whose position can be accessed by the MPC only when it is in its sensing horizon. The data from the two tests are shown in the Fig. 5.5. We can see that the local planner takes into account the obstacles and avoids them optimally. While the obstacles were stationary in the previous experiment, in the next, we tried to analyze how our MPC behaves and controls the car when it encounters a moving obstacle. We command the vehicle to go to a reference target position (x_r, y_r) which was (3,2.5) in this case. The movement of the obstacle was defined by a simple linear function given by Equation 5.14.

$$\begin{aligned} x_{obs}(k+1) = \quad & x_{obs}(k) + \delta t\left[\left(X(k) - x_{obs}(k)\right)\alpha\right] \\ y_{obs}(k+1) = \quad & y_{obs}(k) + \delta t\left[\left(Y(k) - y_{obs}(k)\right)\alpha\right], \end{aligned} \tag{5.14}$$

where (x_{obs}, y_{obs}) and (X, Y) represent the positions of the obstacle and the vehicle respectively. α is a scaling factor which is used to simulate changes in velocity of the obstacle.

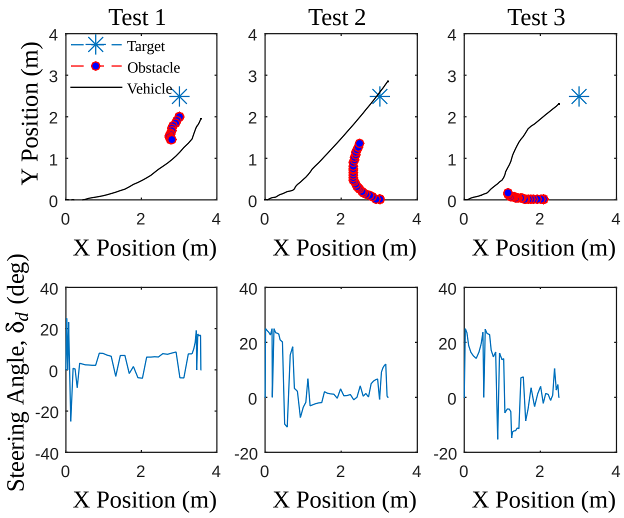


Figure 5.6: Vehicle path trace and steering command with moving obstacle scenarios

In test 1, test 2 and test 3, the starting position of the moving obstacle were (3,2), (3,0) and (2,0) respectively and then they trace the path as shown in Fig. 5.6 towards the scaled car. In all the cases, we assumed that the MPC can anticipate the movement accurately as Equation 5.14 is added as a constraint. In the Test 2 and Test 3 as shown in Fig. 5.6, we can see that the MPC does a good job in avoiding the moving obstacle as well as reaching the target destination optimally. While in Test 1, the MPC does a good job in avoiding the obstacle while managing to be in the vicinity of the target. We will try to analyze and perform experiments on how the MPC behaves when it has limited or partial information about the movement of obstacle in Sec. 6.2.2

Chapter 6

Vehicle-in-the-Loop Platform for Verification of Obstacle Avoidance

A central server has been setup in the lab which enables data storage, information sharing, updating, running global planner and a vehicle-in-loop simulator (simulation environment). The scaled experimental car in conjunction with the server form the vehicle-in-loop platform. The central server also hosts a SQL data server for data storage.

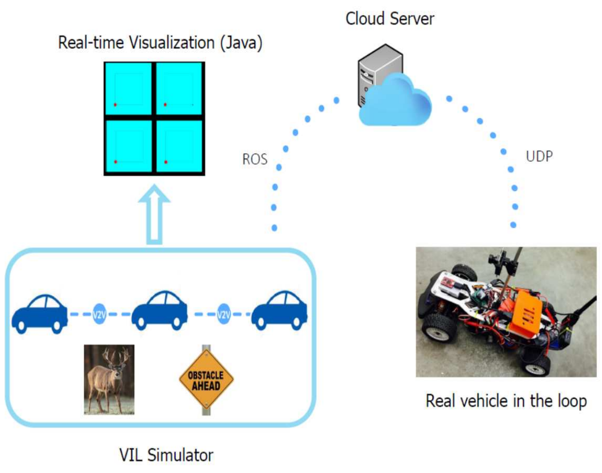


Figure 6.1: Yaw Angle Estimation

As shown in the figure, the VIL simulator runs the virtual vehicle based on the dynamic vehicle model (defined by Equation 4.1) as the plant model and the online MPC (defined by Equation 5.3). It also runs the virtual obstacles and animals. A real-time visualization developed with assistance from Dr. Ali Fayazi, helps to analyze the interaction between the real vehicle and the virtual environment. VIL simulator runs a ROS master and performs all communications via ROS communication protocol. The communication between the VIL simulator and the scaled car is routed via the centralized server which uses UDP encoded as a JSON object over simple Wi-Fi. This helps in sharing information about virtual obstacles, animal and vehicles running on the vehicle-in-loop simulator and the scaled car. In the future all route critical information (elevation, soil, vegetation, obstacle, risk maps, etc.) collected before and during the mission can be stored as multi layered maps, each representing an information layer, in the database. The information stored in the database are accessible by the vehicles on the ground as well as by the global planner which runs on the central computation server. All the maps are stored with a corresponding timestamp representing the latest update. Individual vehicles in the fleet can subscribe/request for the information according to their requirements. Each vehicle publishes its current position, heading and its anticipated control action along with a unique vehicle ID. Any other vehicle within its range will be able to receive the messages and take appropriate corrections to its control action. In this research, we have provided an outline of the communication infrastructure developed while this platform will provide an opportunity to analyze the impact of connectivity in a fleet by studying the interactions between the scaled car and simulated vehicles which will run on the central server, a part of our ongoing research.

6.1 VIL Simulator

The VIL simulator uses dynamic vehicle model as detailed in Sec. 4.2 as the plant model for the simulated vehicles.

Fig. 6.2 details the framework in which the simulated vehicles run. The whole framework runs in the central server. The VIL simulator is also used for simulated obstacles, animals,etc. One of the purposes of building this environment was to analyze and quantify the impact of connectivity on path planning. Another purpose it served was tuning and debugging the MPC weights and PD controllers prior to running it on the scaled car. The global server has information about all the obstacles in our experiments. It is passed on to the scaled car when the obstacle is in the range of the sensing horizon of the car via V2I communication. Our vehicle is not equipped with LIDAR, Stereo Camera or Ultrasonic sensors and hence it is not capable of detecting new obstacles which are not present in the global server. As a future research, we plan to use a combination of the sensors to update the obstacle layer in the global server as discussed in Chapter 2.

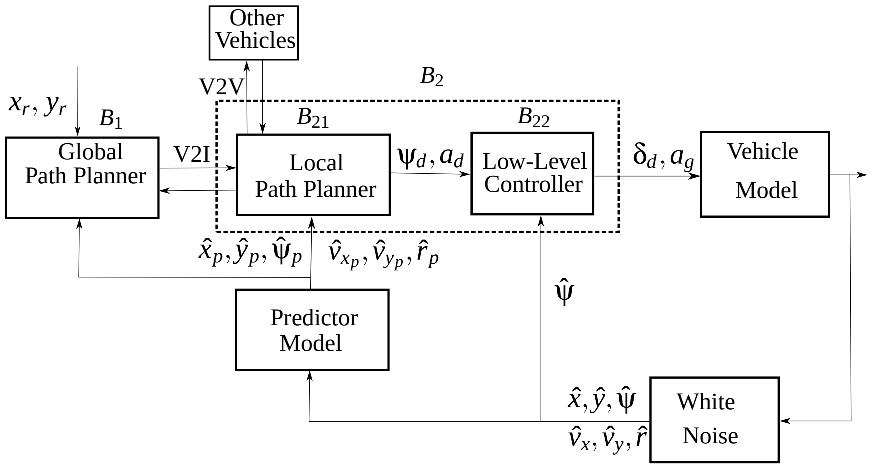


Figure 6.2: Framework adapted for simulated vehicles.

6.2 Experiments: Collision Avoidance with Various Levels of Information and Animal Crossing Accidents

In Section 5.4.2, we have seen how the path planning is effective in avoiding moving obstacles when the MPC has complete information. In the vehicle-in-the-loop environment, we will try to analyze the impact in situations when there is partial or noisy information about approaching vehicles. As a demo example, we will simulate an animal crossing scenario and analyze how the obstacle avoidance algorithms perform under different conditions.

6.2.1 Collision Avoidance with Various Levels of Information

In Fig. 6.3, we try to analyze collision avoidance with approaching vehicles, when the sensors cannot measure the velocity of the obstacle in the co-simulation environment (explained in Sec. 6.1) and compare it with the situation where velocity measurements are available. The goal of the experiment was to reach the target located at (9,9) while avoiding the obstacle on the way and minimizing the cost function defined by Equation 5.4.

In test 1, velocity measurements are available and hence MPC can anticipate the movement more precisely compared to test 2 where the sensors cannot measure the velocity of the obstacle. In this case, it treats the obstacle as stationary at every iteration. In test 1 the MPC is aware about the obstacle movement and hence it makes course correction much before test 2 where it has to suddenly correct its path based on the location of the obstacle. It can also be seen that in test 2, the vehicle has to use more steering effort to

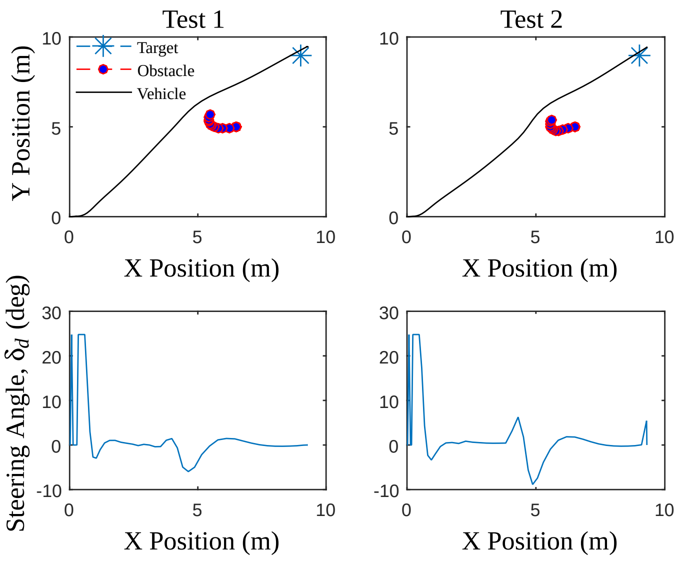


Figure 6.3: Vehicle path trace and steering command using simulation environment in two scenarios: a) MPC has complete information about the approaching obstacles b) MPC is not informed about the velocity of approaching obstacles

avoid the obstacle than in test 1. The condition of sensor information was simulated by adding equation 5.14 to the constraints of the MPC in test 1. This experiment shows how our VIL platform can help in analysis of various scenarios like collision avoidance.

In the next experiment we tried to see the impact of noisy signals on obstacle avoidance. Here, the MPC considers that the obstacle will continue moving with the same velocity as the virtual sensors measure. In Fig. 6.4, test 1 there is no noise in the sensed position of the obstacle and in test 2 there is white noise in the sensed position of the obstacle with standard deviation of 0.25. The goal of the experiment was similar to the previous experiment, i.e to reach the Target at (9,9) while avoiding the obstacle on the way and minimizing the cost function defined by Equation 5.4. The vehicle is able to dodge the moving obstacle since the MPC cost related to the obstacle ensures that the vehicle avoids a circular region around the obstacle. Obstacle$_{\text{real}}$ was the real position traced by the obstacle and Obstacle$_{\text{sensor}}$ was the sensed position traced by the obstacle which included white noise. To improve the obstacle avoidance in case of more noisy signals, vehicle safety distance (d_{safe}) can be increased. Also a probability model can be used for noisy signals which will be a part of our future work.

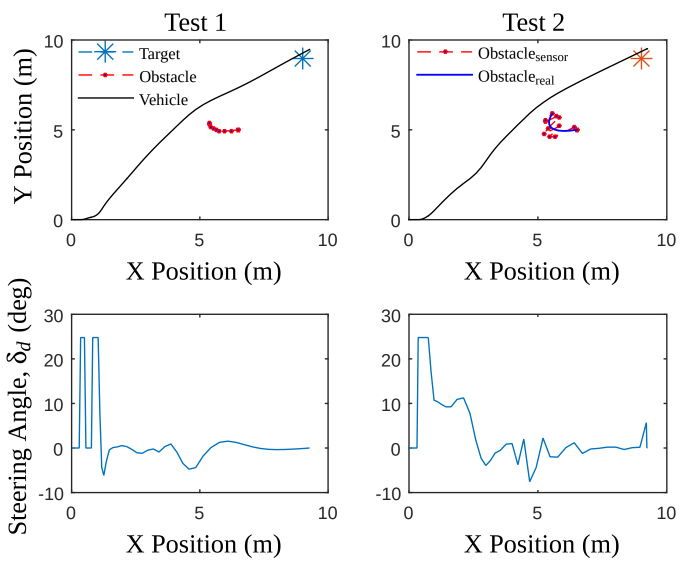


Figure 6.4: Obstacle avoidance with moving obstacle and noise in sensor.

6.2.2 Using VIL to simulate animal-vehicle collisions

Animal-vehicle collisions (AVCs) are a huge challenge specially in wildlife corridors but still little is known about AVCs and what goes wrong in such incidents. Lima et. al. [26] have provided insight into how animals react to oncoming vehicles. As an example we will try a scenario of AVCs. It is a perfect example of benefit of VIL because including a real animal in an experiment is very difficult or impossible. With the proposed method we can simulate the interaction of a real vehicle with virtual animals. In the proposed scenario, we consider that animal behaviours could be predicted accurately based on the the kind of animal, its immediate reactions or movements and conditions of the encounter. We have tried to simulate four probable behaviours animals might exhibit when it encounters a vehicle and try to analyze how our obstacle avoidance algorithms will behave if it had knowledge on the kind of reaction the animal makes. The four scenarios are :

- (a) Blind Crossing : In this condition, we have considered that the animal doesn't notice the vehicle but it can be seen by the sensors on the vehicle.
- (b) Panic Stop : In this condition, we have considered that the animal panics and stops as soon as it notices the vehicle.
- (c) Panic Run : In this condition, we have considered that the animal panics as it notices the vehicle and runs in the direction of its motion.

- (d) Panic Return : : In this condition, we have considered that the animal panics as it notices the vehicle and runs in the opposite direction of its motion.

We initially run our simulations in a software-in-loop environment (SIL) where both the vehicles as well as the animals are simulated run. After, the verifications, we run the scaled-car with simulated animals in a vehicle-in-loop (VIL) environment. To include the predictive movement of the animal in the MPC, following states given by Equation 6.1 are added,

$$\begin{aligned} x_{obs}(k+1) &= x_{obs}(k) + \delta t\left[\left(v_{x_{obs}}(k)\right)\alpha_x\right] \\ y_{obs}(k+1) &= y_{obs}(k) + \delta t\left[\left(v_{y_{obs}}(k)\right)\alpha_y\right], \end{aligned} \tag{6.1}$$

where (x_{obs}, x_{obs}) and $(v_{x_{obs}}, v_{y_{obs}})$ are the position and velocity of the animal, while α_x, α_y are two factors for incorporating the predictive movement of the animal given by,

$$\begin{aligned} \alpha_x &= \alpha \cos\left(\tan^{-1}\left(\frac{v_{y_{obs}}}{v_{x_{obs}}}\right)\right) \\ \alpha_y &= \alpha \sin\left(\tan^{-1}\left(\frac{v_{y_{obs}}}{v_{x_{obs}}}\right)\right) \end{aligned} \tag{6.2}$$

$$\alpha = \begin{cases} 1 & \text{if scenario is blind crossing} \quad \text{or} \quad d_{obs} > d_{safe} \quad \text{or} \quad v_{m_{obs}} = v_{obs} \\ 0 & \text{if scenario is panic stop} \quad \text{and} \quad d_{obs} \leq d_{safe}, \\ \eta & \text{if scenario is panic run} \quad \text{and} \quad d_{obs} \leq d_{safe} \quad \text{and} \quad v_{m_{obs}} > v_{obs}, \\ -\eta & \text{if scenario is panic run} \quad \text{and} \quad d_{obs} \leq d_{safe} \quad \text{and} \quad v_{m_{obs}} > v_{obs}, \end{cases} \tag{6.3}$$

where $v_{m_{obs}}$ and v_{obs} are the maximum and the measured speed of the animal, d_{obs} and d_{safe} are the measured and safe distance to the animal, and η is the ratio of maximum to the current speed of the animal.

Fig. 6.5 shows the four scenarios simulated in a software-in-loop environment while Fig. 6.6 shows them in a vehicle-in-loop environment. The green vectors indicate the relative position of the vehicle and the animal. In scenario (a) and scenario (c), we can see that the vehicle waits for the animal to pass, while in scenario (b) and scenario (d) the MPC knows that the animal is either going to stop or change its direction and hence makes the necessary control decisions to avoid a collision.

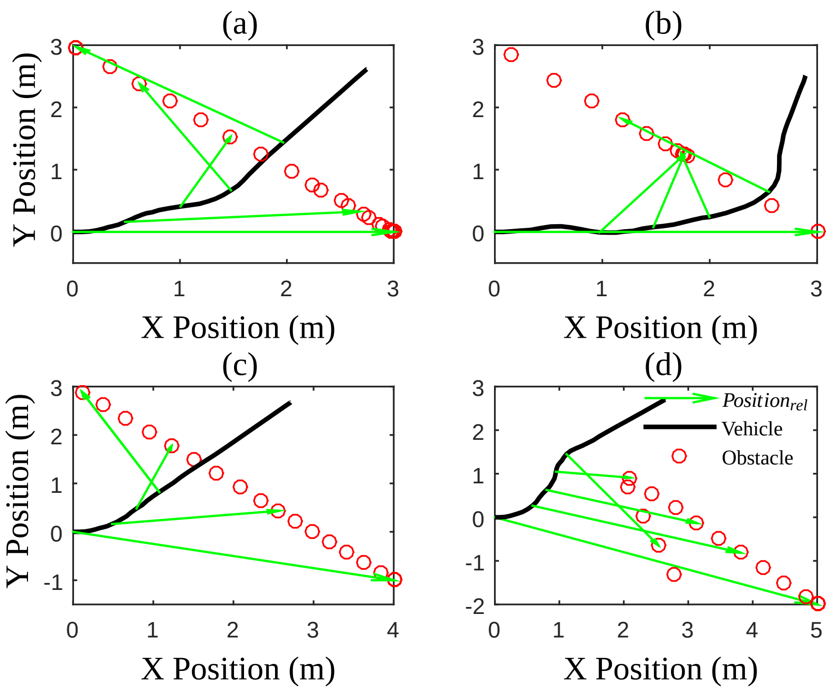


Figure 6.5: Analysis of animal-vehicle collision avoidance using software-in-loop in four scenarios: a) Blind Crossing, b) Panic Stop, c) Panic Run and d) Panic Return

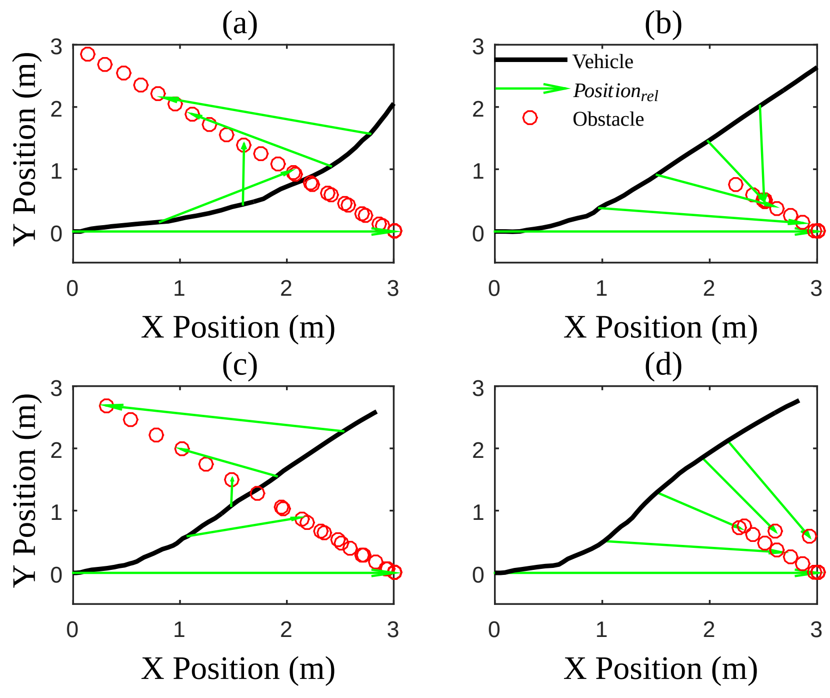


Figure 6.6: Analysis of animal-vehicle collision avoidance using vehicle-in-loop in four scenarios: a) Blind Crossing, b) Panic Stop, c) Panic Run and d) Panic Return

Chapter 7

Conclusions

The first stage of this research was conducted in collaboration with Dr. Judhajit Roy and Dr. Nianfeng Wan where we formalized path planning for off-road terrain as a two stage optimal control problem: The first stage relies on large scale but perhaps coarse multi-layer maps and employs dynamic programming to calculate the optimal cost-to-go from any point on the coarsely gridded map to a pre-specified destination. The second stage is based on MPC which calculates the optimal path in a receding horizon, centered on a vehicle, and based on higher resolution data from on-board sensors, guided by the cost-to-go map. The layer based approach helped in efficient handling of route critical information as shown in our experiment where, if a vehicle is not informed about enemy's Line of Sight and doesn't take it into account, it might get trapped. Using a centralized server for storing all the route information which can be enriched by the vehicles also ensures that all vehicles in the platoon have access to the latest information.

While the first stage of the research was carried out in a simulation environment, the the next stage focused on experimental implementation of the route guidance framework in a scaled platform. The efficacy of the localization, obstacle avoidance and vehicle control strategies developed have been shown experimentally, specifically the drift correction algorithm for the IMU. An important aspect of the research is the utilization of a low-level controller for augmenting the control decisions made by the MPC with minimal computational resources available. Online measurement of steering angle using a camera-based tire edge detection algorithm as well as steering angle estimation were other novel aspects of this research. Due to the constraints on the computational power, we couldn't implement the global cost-to-go map in the local planner but it is in the scope of future work of this project.

Finally, the vehicle-in-the-loop framework helped in analysis of our obstacle avoidance algorithms in various scenarios. In the animal-vehicle collision simulations, we demonstrated a beneficial instance of VIL in an animal avoidance scenario where experimenting with a real animal is not practical.

Future scope of this work include the following:

- use a powerful on-board computer which will enable to activation of online steering angle measurement algorithm along with the MPC for more robust steering control.
- use higher order models starting with the kinetic model used for state estimation in the MPC, so that it can take into account the dynamic forces working on the vehicle.
- run the vehicle in off-road terrain with different information layers like elevation, soil, visibility and use the cost-to-go map to guide the local planner.
- use the vehicle-in-the-loop platform for analyzing the impact of connectivity on fleet of vehicles, where each vehicle can update its position, the perceived terrain and soil information to the server which helps to create a dynamic map of any unexplored terrain. This will enable successive vehicles in the fleet to learn from the mistakes of the preceding vehicles and gradually move faster, safer, and more energy efficiently. Eventually, this test bed will aid future research in the domain of autonomous vehicles for preliminary algorithm verification like collision avoidance and performing feasibility tests.
- use of combination of sensors like LIDAR, stereo camera, etc for online obstacle detection and updating in the global obstacle map layer.

Bibliography

[1] http://nationalmap.gov/3dep/.

[2] https://barc-project.com.

[3] https://nationalmap.gov.

[4] https://websoilsurvey.sc.egov.usda.gov.

[5] Motilal Agrawal and Kurt Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 1063–1068. IEEE, 2006.

[6] US Army. *FM 5-430-00-1 AFJPAM 32-8013, Vol 1 Planning and Design of Roads, Airfields, and Heliports in the Theater of Operations–Road Design*. Department of the Army (US), 1994.

[7] Alberto Broggi, Elena Cardarelli, Stefano Cattani, and Mario Sabbatelli. Terrain mapping for off-road autonomous ground vehicles using rational b-spline surfaces and stereo vision. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 648–653. IEEE, 2013.

[8] S Campbell, Wasif Naeem, and George W Irwin. A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres. *Annual Reviews in Control*, 36(2):267–283, 2012.

[9] Steve Carver and Justin Washtell. Real-time visibility analysis and rapid viewshed calculation using a voxel-based modelling approach. In *GISRUK 2012 Conference, Apr*, volume 11, 2012.

[10] Jaewoong Choi, Junyoung Lee, Dongwook Kim, Giacomo Soprani, Pietro Cerri, Alberto Broggi, and Kyongsu Yi. Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):974–982, 2012.

[11] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[12] Christer Ericson. *Real-time collision detection*. CRC Press, 2004.

[13] Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

[14] J Gonzales, F Zhang, K Li, and F Borrelli. Autonomous drifting with onboard sensors. In *Advanced Vehicle Control: Proceedings of the 13th International Symposium on Advanced Vehicle Control (AVEC'16), September 13-16, 2016, Munich, Germany*, page 133. CRC Press, 2016.

[15] Héctor H González-Banos and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

[16] A. Goswami, H. Saeidi, A. Vahidi, and Jayakumar P. Implementation of hierarchical framework with accurate localization and low level control in a low-cost scaled autonomous car, in review. *American Control Conference (ACC), 2018*.

[17] Rami Y Hindiyeh. Dynamics and control of drifting in automobiles. *(Ph.D. thesis), Stanford University*, 2013.

[18] Larry D Jackel, Eric Krotkov, Michael Perschbacher, Jim Pippine, and Chad Sullivan. The darpa lagr program: Goals, challenges, methodology, and phase i results. *Journal of Field Robotics*, 23(11-12):945–973, 2006.

[19] E Jelavic, J Gonzales, and F Borrelli. Autonomous drift parking using a switched control strategy with onboard sensors. *IFAC-PapersOnLine*, 50(1):3714–3719, 2017.

[20] Arno Kamphuis, Michiel Rook, and Mark H Overmars. Tactical path finding in urban environments. In *First International Workshop on Crowd Simulation*, 2005.

[21] Bryan A Knowles. In the face of anticipation: Decision making under visible uncertainty as present in the safest-with-sight problem. 2016.

[22] Hung Manh La, Ronny Salim Lim, Jianhao Du, Sijian Zhang, Gangfeng Yan, and Weihua Sheng. Development of a small-scale research platform for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1753–1762, 2012.

[23] Gerhard Lakemeyer and Bernhard Nebel. *Exploring artificial intelligence in the new millennium*. Morgan Kaufmann, 2003.

[24] Jean-François Lalonde, Nicolas Vandapel, Daniel F Huber, and Martial Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of field robotics*, 23(10):839–861, 2006.

[25] Ping Li, Junyan Zhu, and Fang Peng. Comparison of a* and lambda* algorithm for path planning. In *Engineering Technology and Applications: Proceedings of the 2014 International Conference on Engineering Technology and Applications (ICETA 2014), Tsingtao, China, 29-30 April 2014*, page 171. CRC Press, 2014.

[26] Steven L Lima, Bradley F Blackwell, Travis L DeVault, and Esteban Fernández-Juricic. Animal reactions to oncoming vehicles: a conceptual review. *Biological Reviews*, 90(1):60–76, 2015.

[27] Jan Marian Maciejowski. *Predictive control: with constraints*. Pearson education, 2002.

[28] Roberto Manduchi, Andres Castano, Ashit Talukder, and Larry Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous robots*, 18(1):81–102, 2005.

[29] Erik Narby. Modeling and estimation of dynamic tire properties, Masters Thesis, 2006.

[30] Hans Pacejka. *Tire and vehicle dynamics*. Elsevier, 2005.

[31] EB Pacis, B Sights, G Ahuja, G Kogut, and HR Everett. An adaptive localization system for outdoor/indoor navigation for autonomous robots. Technical report, SPACE AND NAVAL WARFARE SYSTEMS CENTER SAN DIEGO CA, 2006.

[32] Sakda Panwai and Hussein Dia. Comparative evaluation of microscopic car-following behavior. *IEEE Transactions on Intelligent Transportation Systems*, 6(3):314–325, 2005.

[33] Yadollah Rasekhipour, Amir Khajepour, Shih-Ken Chen, and Bakhtiar Litkouhi. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1255–1267, 2017.

[34] J. Roy, N. Wan, A. Goswami, A. Vahidi, P. Jayakumar, and C. Zhang. A hierarchical route guidance framework for off-road connected vehicles, in review. *ASME Journal of Dynamic Systems, Measurement, and Control*.

[35] Zvi Shiller and Y-R Gwo. Dynamic motion planning of autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 7(2):241–249, 1991.

[36] David Ezra Sidran. Good decisions under fire.

[37] Moshe Sniedovich. Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and cybernetics*, 35(3):599, 2006.

[38] David Stavens and Sebastian Thrun. A self-supervised terrain roughness estimator for off-road autonomous driving. *arXiv preprint arXiv:1206.6872*, 2012.

[39] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[40] William E Travis, Randy J Whitehead, David M Bevly, and George T Flowers. Using scaled vehicles to investigate the influence of various properties on rollover propensity. In *American Control Conference, 2004. Proceedings of the 2004*, volume 4, pages 3381–3386. IEEE, 2004.

[41] Rajeev Verma, Domitilla Del Vecchio, and Hosam K Fathy. Development of a scaled vehicle with longitudinal dynamics of an hmmwv for an its testbed. *IEEE/ASME Transactions on Mechatronics*, 13(1):46–57, 2008.

[42] Qiannan Wang and Steffen Müller. A hierarchical controller for path planning and path following based on model predictive control. In *Advanced Vehicle Control: Proceedings of the 13th International Symposium on Advanced Vehicle Control (AVEC'16), September 13-16, 2016, Munich, Germany*, page 195. CRC Press, 2016.